OGPO: Sample Efficient Full-Finetuning of Generative Control Policies

Sarvesh Patil[a,$], Mitsuhiko Nakamoto[b,$], Manan Agarwal[a,‡], Shashwat Saxena[a,‡], Jesse Zhang[c,‡], Giri Anantharaman[a], Cleah Winston[c], Chaoyi Pan[a], Douglas Chen[a], Nai-Chieh Huang[a], Zeynep Temel[a], Oliver Kroemer[a], Sergey Levine[b], Abhishek Gupta [cd], Hongkai Dai[d,†], Paarth Shah[d,†], Max Simchowitz[a,†]

[$]Project lead. ‡Equal Contribution. [†]Equal advising.

[a]Carnegie Mellon University [b]University of California, Berkeley [c]University of Washington

[d]Toyota Research Institute

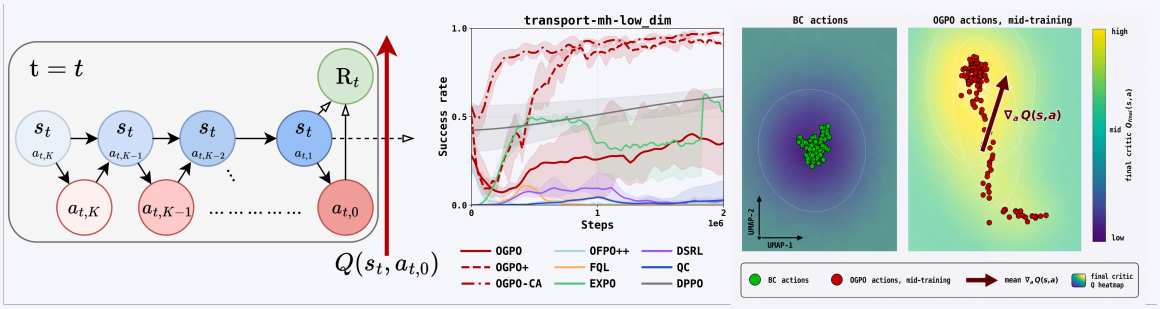


Figure 1: OGPO enables sample-efficient full-policy finetuning of generative control policies. Left: A generative control policy (GCP) represents action generation as a sequential computation, constituting a *denoising MDP* at each environment step. Whereas prior work [Ren et al., 2024], embeds the denoising MDP into the environment to form a bi-level MDP, OGPO severs this connection, and instead maximizes an off-policy critic as a terminal reward via PPO-style optimization over the denoising trajectories. **Middle:** We show that OGPOsubstantially improves sample efficiency on challenging manipulation tasks like ROBOMIMIC TRANSPORT compared to prior GCP finetuning methods, even with limited hyperparameter tuning. **Right:** Despite its sample efficiency, OGPO preserves non-trivial variance in the action distribution, preserving the capacity for exploration. As show, action variance is "squeezed" to be *perpendicular to critic gradients* during the middle of the training runs. The critic is insensitive to variance in these actions, so action variation does not conflict with high performance.

[a]{sarveshp, mananaga, ssaxena2, dchen3, chaoyip, naichieh, giria, msimchow}@andrew.cmu.edu
[b]{nakamoto, svlevine}@eecs.berkeley.edu
[c]{jessezha, cleahw, abhgupta}@cs.washington.edu
[d]{hongkai.dai,paarth.shah}tri.global

Code: https://github.com/simchowitzlabpublic/OGPO_public

Date: June 15, 2026

Abstract

Generative control policies (GCPs), such as diffusion- and flow-based control policies, have emerged as effective parameterizations for robot learning. This work introduces Off-policy Generative Policy Optimization (OGPO), a sample-efficient algorithm for finetuning GCPs that maintains off-policy critic networks to maximize data reuse and propagate policy gradients through the full generative process of the policy via a modified PPO objective, using critics as the terminal reward. OGPO achieves state-of-the-art performance on manipulation tasks spanning multi-task settings, high-precision insertion, and dexterous control. To our knowledge, it is also the only method that can *fine-tune poorly-initialized behavior cloning policies to near full task-success with no expert data in the online*

replay buffer, and does so with *few task-specific hyperparameter tuning*. Through extensive empirical investigations, we demonstrate that **OGPO** drastically outperforms methods alternatives on policy steering and learning residual corrections, and identify the key mechanisms behind its performance. We further introduce practical stabilization tricks, including success-buffer regularization, two-sided conservative advantages, and Q-variance reduction, to mitigate critic over-exploitation across state- and pixel-based settings. Beyond proposing **OGPO**, we conduct a systematic empirical study of GCP finetuning, identifying the stabilizing mechanisms and failure modes that govern successful off-policy full-policy improvement.

1 Introduction

Autonomous acquisition of new skills is an important challenge for modern robot manipulation. While imitation learning via behavior cloning (BC) from human demonstration can enable a robot to learn behaviors across several contexts, performance is typically brittle to subtle changes in tasks and environments. These models rarely exhibit high success rates zero-shot in the diversity of settings encountered in deployment. While this fragility can be remedied through additional data collection, a natural question to ask is - can the robustness of pre-trained imitation learning policies be bolstered autonomously without requiring considerably more manual data collection?

To this end, there has been a strong interest in finetuning pre-trained robotic policies via reinforcement learning (RL), to autonomously improve behavior via self-collected experience. Of particular relevance is the problem of finetuning *Generative Control Policies* (GCPs) - the parametrization of control policies by expressive generative models, such as diffusion or flow models [Chi et al., 2023, Black et al., 2024, Pan et al., 2025]. These policies have been extremely effective for modern robotic applications [Zhang and Gienger, 2024, Wolf et al., 2025].

Current methodology for GCP finetuning succumbs to tradeoffs between data efficiency and the extent of policy improvement during training. Approaches focused on sample efficiency combine *off-policy* critic learning, enabling strong experience reuse, with either targeted *partial* finetuning of the GCP, such as steering the initial generation noise or learning residual corrections, or instead use behavior cloning to imitate high-return actions. These approaches learn quickly when the base policy has strong coverage of optimal actions, but struggle with exploring new behavior. On the other hand, methods focused on eliciting maximum final task performance [Lei et al., 2025, Ren et al., 2024, McAllister et al., 2025] use *on-policy* policy gradient updates, which drive aggressive policy improvement at the expense of significantly compromised sample efficiency.

In this work, we propose a new algorithm - **OGPO** for full-finetuning of expressive GCPs, providing both sample-efficient and expressive policy updates via data-efficient off-policy reinforcement learning. Following Ren et al. [2024], Black et al. [2023], **OGPO** views GCP optimization as a bi-level MDP, with a nested inner denoising MDP over the action generation steps of a GCP, and an outer environment dynamics MDP over actions actually executed in the environment. Importantly, in real-robotics tasks, there exists a *sample-cost asymmetry*: collecting trajectories from the environment MDP is expensive, while generating action trajectories through the denoising MDP is purely computational and therefore cheap.

While direct policy optimization in the unrolled bi-level MDP can be very (environment-)sample inefficient [Ren et al., 2024, Zhang et al., 2025], **OGPO** leverages the aforementioned sample-cost *asymmetry* to perform *decoupled* policy optimization. Specifically, **OGPO** performs sample-efficient, off-policy Temporal Difference (TD)-learning to learn a Q function in the environment dynamics MDP over *expensive* environment samples, while using data-inefficient but stable on-policy RL updates to extract policies from the inner denoising MDP over *cheap* GCP samples (see Figure 1 left). Doing so allows for an off-policy policy optimization algorithm that is data-efficient (due to TD-learning in the environment dynamics MDP), yet expressive (due to on-policy RL finetuning in the denoising MDP)

Through careful empirical study, we show that **OGPO** is able to achieve both stable and expressive updates for finetuning GCPs in challenging robotics tasks. Based on empirical analysis of the shortcomings, we further propose **OGPO+**, an empirically optimized variant that incorporates improvements in test-time optimization such as Best-of-N planning via Q-functions and policy distillation from successful trajectories obtained via online RL. These improvements allow **OGPO+** to achieve state-of-the-art performance on a set of contact-rich simulation environments with varying horizons, degrees of freedom, and precision requirements, while requiring minimal hyperparameter tuning. Surprisingly, we show that **OGPO+** is able to fine-tune policies with *zero expert data* in the policy replay buffer. This is a fundamentally new capability that points towards the future possibility of finetuning models with minimal human data collected in a task-specific manner on deployment. We perform a careful set of analyses to understand the impact of the decoupled optimization central to **OGPO**, and the impact of the design decision made in **OGPO+** - showing the efficacy of full-policy finetuning of GCPs under the right design choices.

2 Preliminaries

We formulate our algorithm as a *Markov Decision Process* (MDP) $M_{\text{ENV}} := (S, A, P_0, P, R, \gamma)$, with states $s \in S$, actions $a \in A$, initial state distribution P_0, transition probabilities P, reward R, and discount factor $\gamma \in (0,1)$. At each timestep t, the agent (e.g., robot) observes the state $s_t \in S$, takes an action $a_t \sim \pi(a_t \mid s_t) \in A$, transitions to the next state s_{t+1} according to $s_{t+1} \sim P(s_{t+1} \mid s_t, a_t)$ while receiving a reward $R(s_t, a_t)$.[1] For the MDP M_{ENV}, we let $\mathbb{E}^{\pi}$ (resp. $\mathbb{P}^{\pi}$) denote the expectation (resp. probability distribution) over trajectories $(s_0, a_0, \ldots, s_T, a_T)$ with length $T+1$, with initial state distribution $s_0 \sim P_0$ and transition operator P. We train a policy to optimize the cumulative discounted return $J(\pi_\theta) = \mathbb{E}^{\pi_\theta}\left[\sum_{t \geq 0} \gamma^t R(s_t, a_t)\right]$. We also recall the Q-function

$$Q^{\pi}(s,a) := \mathbb{E}^{\pi}[\sum_{t \geq 0} \gamma^t R(s_t, a_t) \mid (s_t, a_t) = (s,a)] \tag{2.1}$$

and value function $V^{\pi}(s) := \mathbb{E}_{a \sim \pi(s)}[Q^{\pi}(s,a)]$. We apply action chunking [Zhao et al., 2023], where sequences of actions $a_{t:t+h-1}$ are predicted and executed in open-loop. For simplicity, we treat each action chunk as a single action in M_{ENV}, thereby preserving the standard MDP notation. Thus, for the rest of the paper, a_t **refers to an entire action-chunk**, and rewards are adjusted appropriately (see Appendix A.1 for how).

On-Policy Policy Gradient Methods. *Policy gradient* (PG) methods (e.g., REINFORCE [Williams, 1992]) improve policy performance by approximating the gradient of this objective w.r.t. the policy parameters:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}^{\pi_\theta}\left[\sum_{t \geq 0} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, r_t(s_t, a_t)\right], \tag{2.2}$$

where $r_t(s_t, a_t) := \sum_{\tau \geq t} \gamma^\tau R(s_\tau, a_\tau)$ is the discounted future return from time t, and $\nabla_\theta \log \pi_\theta(a_t | s_t)$ denotes the gradient of the logarithm of the *likelihood* of $(a_t \mid s_t)$. Myriad improvements exist to reduce variance of gradient estimation and accelerate training stability; following [Ren et al., 2024, Zhang et al., 2025], we build on the PPO algorithm [Schulman et al., 2017]. PG methods are called *on-policy* because they optimize over the *current* policy distribution, limiting data re-use and sample efficiency.

[1] In practice, algorithms may be given incomplete or redundant state observation (e.g., via pixel measurements), in which case we can replace s with an observation o. This may violate the Markovian condition in the MDP, but still leads to well-posed algorithms.

Off-Policy Reinforcement Learning. *Off-policy RL methods* maintain a long horizon replay buffer $\mathscr{D}_{\text{roll}} = \{(s_t, a_t, s_{t+1}, r_t, d_t)\}$ consisting of past states s_t, actions a_t, subsequent states s_{t+1} from the environment transitions, the observed rewards r_t, and the done signal d_t. The buffer is used to train an ensemble of M critic networks $Q_{\phi_i} : S \times A \to \mathbb{R}$, with parameters $\phi_1, \ldots, \phi_M$, such that $Q_{\phi_i}(s_t, a_t)$ evaluates the expected cumulative return $Q^{\pi_{\bar{\theta}}}(s_t, a_t)$ of action a_t at state s_t under a current *target policy* $\pi_{\bar{\theta}}$. The critic networks are updated in parallel using the temporal difference loss, which enforces the Bellman consistency equation defined by Q-functions:

$$L_{\text{critic}}(\phi) = \mathbb{E}\Big[Q_\phi(s_t, a_t) - \Big(r_t + \gamma \cdot Q_{\text{targ}}(s_{t+1}, a_{t+1})\Big)\Big]^2, \tag{2.3}$$

where above the expectation $\mathbb{E}$ is taken over $(s_t, a_t, r_t, s_{t+1} \sim \mathscr{B})$ sampled from the replay buffer $\mathscr{B}$, and each a_{t+1} is sampled independently from the current target policy $\pi_{\bar{\theta}}(\cdot \mid s_{t+1})$. To avoid overestimation bias, we set $Q_{\text{targ}}(s, a) = \frac{1}{M}\sum_i Q_i$ to be a mean over critic networks, described in the Appendix (Appendix A.1). [Fujimoto et al., 2018, Chen et al., 2021].Importantly, (2.3) enables data collected by policies from previous training epochs, thereby increasing sample efficiency.

Generative Control Policies. Current robotic control policies use generative models as parameterizations of control policies. Following [Pan et al., 2025], we call these generative control policies (GCPs). GCPs represent a stochastic policy $\pi_\theta(\cdot \mid s)$ as a series of iterative computation steps, defined by a mapping $\bar{\pi}_\theta : S \times A \times \mathbb{N}$. Given a state s_t, the policy first samples $a_{t,K} \sim \bar{\pi}_\theta(\cdot \mid a_{t,k} = \emptyset, k = K, s_t)$ where k is a GCP timestep. Next, we sample $a_{t,k-1} \sim \bar{\pi}_\theta(\cdot \mid a_{t,k}, k, s_t)$ which leads to is an action $a_{t,0}$. We compactly denote the distribution of this action given the observation as $a_{t,0} \sim \pi_\theta(\cdot \mid s_t)$, turning the GCP into a standard policy. Our iteration conventions are *decreasing* in K, those in diffusion models. Following the same conventions, we also refer to the index k as the "denoising step."

Flow-Based GCPs. We focus on a popular class of GCPs: flow-based control policies [Black et al., 2024]. As discussed in Appendix C, our methods and baselines can also be instantiated with Diffusion-based policies [Chi et al., 2023] and other controller parameterizations [Pertsch et al., 2025, Frans et al., 2024, Pan et al., 2025] . Flow policies are pretrained using the flow-matching objective: given training pairs (s, a), we sample noise $z \sim \mathrm{N}(0, \mathbf{I})$. With a continuous noise index $\tau \in [0, 1]$, we define an interpolated action $a_{(\tau)} = \tau + (1 - \tau)z$, and optimize a velocity field $v_\theta(a_{(\tau)}, \tau; s)$ by minimizing $\mathbb{E}_{(s,a,\tau)}\|v_\theta(a_{(\tau)}, \tau; s) - (a - z)\|^2$ [Albergo et al., 2023, Lipman et al., 2022] . Inference is performed by discretizing an ordinary differential equation (ODE) which reverses the noising process $a_{t,k-1} := a_{t,k} + \frac{1}{K} v_\theta(a_{t,k}, k/K, s)$, with $a_{t,0} \sim \mathrm{N}(0, \mathbf{I})$.

3 Off-Policy Generative Policy Optimization

We propose **O**ff-Policy **G**enerative **P**olicy **O**ptimization, **OGPO**, an off-policy full-policy finetuning method for generative control policies. We begin by introducing the basic algorithm, and then describe an improved variant, **OGPO+**. We provide summary pseudocode in Algorithm 1, and defer full implementation details to Appendix B.

Background: Off-Policy Policy Extraction. Given a replay buffer $\mathscr{B} = \{(s, a, s', r)\}$, traditional off-policy RL methods consist of two steps: (1) fitting Q- functions via a TD-update Eq. (2.3), (2) performing policy extraction by choosing actions that maximize the target Q function Q_{targ} as a surrogate of future return:

$$\theta \in \arg\max_\theta \mathbb{E}_{a \sim \pi_\theta(s)}(Q_{\text{targ}}(s, a)). \tag{3.1}$$

The replay buffer facilitates off-policy data-reuse for training Q_{targ} (typically via (2.3)), driving sample efficiency, whereas (3.1) can be computed purely computationally. A historically popular approach to optimize this objective for simple policy parametrization, like Gaussian policies, is the so-called *reparameterization trick* [Kingma and Welling, 2013, Figurnov et al., 2018], where a stochastic policy is rendered as $\pi_\theta(s; w)$ for a noise w drawn from a fixed (non-learned) distribution. From here, Eq. (3.1) is written as an expectation over w, the algorithm directly differentiates $Q_{\text{targ}}(s, \pi_\theta(s, w))$ with respect to θ under samples w. In principle, the same can be done for GCPs such as flow-policies, sampling an initial noise $a_{t,K}$ and backpropagating through the inference chain (Figure 4, center). However, as we show experimentally (Appendix H.1), doing so leads to an exploding gradient problem as we differentiate through the multiple flow steps. Moreover, it requires differentiating $\nabla_a Q_{\text{targ}}$, which can be inaccurate in contact-rich tasks [Suh et al., 2022].

OGPO: On-Policy PPO for Off-Policy Policy Extraction. **OGPO** is designed for applications, such as robotic manipulation, where environment interactions are more costly than computation, and where action gradients with respect to Q_{targ} are noisy or inaccurate [Suh et al., 2022]. We maintain off-policy critic learning that facilitates data reuse, and propose a *fully parallelizable zero-order optimizer* that solves Eq. (3.1), avoiding backpropagation through the denoising chain and differentiation with respect to the target network.

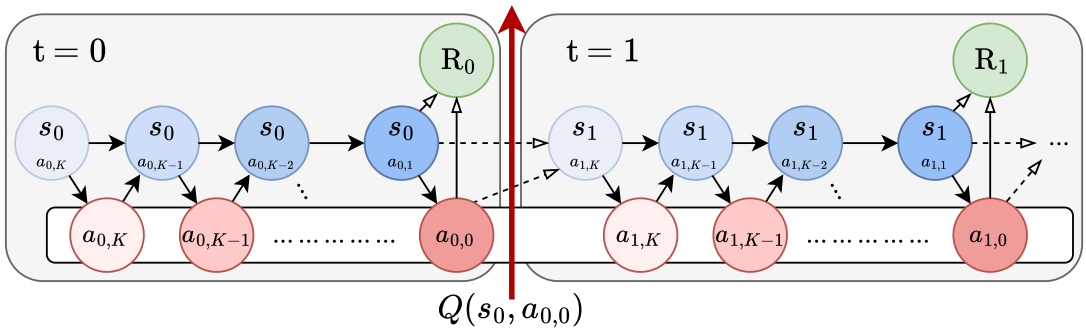


Figure 2: We recall the bi-level MDP from [Ren et al., 2024], which embeds action-level trajectories into the environmental dynamics. **OGPO** truncates this MDP at the end of each denoising trajectory, using Q-values as a terminal, action-trajectory-level reward, enabling off-policy policy extraction via on-policy policy optimization.

Our starting point is the bi-level MDP formulation adopted from [Ren et al., 2024] (Figure 2). Following [Black et al., 2023], we view sequences $a_{t,K:0} = (a_{t,K}, \ldots, a_{t,0})$ as trajectories in an *denoising* MDP, where time is indexed by denoising step k, and state and action at step k are $a_{t,k}$ and $a_{t,k-1}$, respectively. Ren et al. [2024] embeds this action-level MDP into the environment-level MDP M_{ENV}, resulting in an *bi-level* MDP where states are $\bar{s}_{t,k} = (s_t, a_{t,k})$, the actions are $a_{t,k-1}$, and the indices (t, k) are lexicographically increasing in t and decreasing in k. Figure 2 depicts this bi-level MDP: transitions within each gray block occur within the denoising-level MDP, and between gray blocks are transitions in M_{ENV}; see Appendix D for further details. The **DPPO** algorithm proposed by [Ren et al., 2024] then applies on-policy PPO at the level of this bi-level MDP. Whilst avoiding the aforementioned pathologies associated with backpropagation, this method gives up the sample efficiency afforded by off-policy critic learning.

Our **key insight** is that denoising-trajectories can be generated purely *computationally* from policy inference, as they occur in the "imagination" of the GCP. We can then use critic learning to sever the bi-level MDP just before environment-MDP state transitions (red line, Figure 2), enabling zero-order optimization applied only to the denoising-level MDP. As compared to backpropagation approaches to

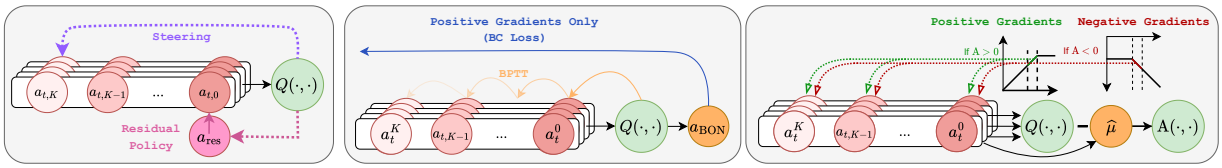


Figure 3: Visual depiction of the different off-policy RL algorithms. **(left)** DSRL trains an initial noise steering policy, while EXPO trains a residual policy to modify final GCP actions. **(center)** QC drives policy improvement via supervised finetuning (SFT) of Best-of-N actions ranked via the critic, while BPTT backpropagates the gradients through the entire GCP. **(right)** OGPO uses an ensemble of critics to compute $\hat{A}^{\mathrm{G}}$ (Eq. (3.2)) that update the GCP via Annealed Importance Sampling, thereby directly conditioning the log-likelihoods over the GCP chain.

solving Eq. (3.1), our approach avoids (i) backpropagation through time and (ii) differentiating through the Q-function. Moreover, as compared to pure on-policy zero order optimization through the bi-level MDP [Ren et al., 2024], our zero-order updates are (i) performed purely computationally, in the "imagination" of the denoising process (ii) fully parallelized across large batch sizes (iii) used to optimize a critic network, facilitating full reuse of environment-level trajectories. Moreover, (iv) the problem horizon of the denoising-level MDP scales only with the denoising steps K, and not $K \times$ (task horizon). Concretely, we apply the PPO algorithm [Schulman et al., 2017], a zero-order policy gradient method, to optimize over the denoising MDP. Given state s_t, denoising trajectory $a_{t,K:0}$, and baseline value estimate $\hat{V}$, we apply the standard PPO loss *only* to the denoising trajectory $a_{t,K:0}$:

$$\ell_{\mathrm{PPO}}(\theta; s_t, a_{t,K:0}, \hat{\mu}) := \min(\omega_\theta \hat{A}, \mathrm{clip}(\omega_\theta, 1-\epsilon, 1+\epsilon)\hat{A})$$
$$\omega_\theta := \prod_{k=1}^{K} \frac{\pi_\theta(a_{t,k-1} \mid s_t, a_{t,k})}{\pi_{\bar{\theta}}(a_{t,k-1} \mid s_t, a_{t,k})}, \; \hat{A} = Q_{\mathrm{targ}}(s_t, a_{t,0}) - \hat{V}. \tag{3.2}$$

Multiple Denoising-Trajectory Sampling. Because denoising-trajectories are generated computationally, they can be resampled *fully in parallel* from *any* given state s_t in the replay buffer. Moreover, Q_{targ} can be evaluated without taking a single transition step in the environment. Taking advantage of this, we evaluate our PPO loss over an average of a batch of parallel-sampled trajectories, purely in the "imagination" of the GCP. By analogy to policy optimization in large language models (LLMs), we can think of a state s_t in the buffer as a "context" and the denoising trajectory $a_{t,K:0}$ as a "response". We draw inspiration from the GRPO algorithm [Shao et al., 2024] in LLM post-training, where multiple responses are sampled in parallel from a given prompt, and gradients are averaged together to reduce gradient variance.[2] In OGPO, at each update, we sample $N_{\mathtt{batch}}$ states $(s^{(i)})_{1 \le i \le N_{\mathtt{batch}}}$ from our replay buffer, and sample $N_{\mathtt{group}}$ denoising trajectories $(a_{K:0}^{(i,j)})_{1 \le j \le N_{\mathtt{group}}}$ drawn i.i.d. from $\pi_{\bar{\theta}}(\cdot \mid s^{(i)})$ per state. We then update via the loss

$$\hat{L}_{\mathrm{PPO}}(\theta) = \frac{1}{N_{\mathtt{tot}}} \sum_i \sum_j \ell_{\mathrm{PPO}}(\theta; s^{(i)}, a_{K:0}^{(i,j)}, \hat{V}^{(i)}). \tag{3.3}$$

Eq. (3.3) averages both over the states $s_t^{(i)}$ from the buffer ("prompts"), and denoising-trajectories generated in parallel from each given state ("responses"). This yields a normalization factor of $N_{\mathtt{tot}} := N_{\mathtt{batch}} \cdot N_{\mathtt{group}}$. Moreover, parallel sampling facilitates estimating the value baseline via a direct Monte-Carlo approximation $\hat{V}^{(i)} \leftarrow \frac{1}{N_{\mathtt{group}}} \sum_j Q_{\mathrm{targ}}(s^{(i)}, a_0^{(i,j)})$, obviating the need to learn a separate value-prediction network.

[2] GRPO includes an additional variance normalization term, which we omit.

Debiasing Noise Injection for Flow Policies. We instantiate OGPO for flow-based policies. To evaluate the likelihood ω_θ in Eq. (3.2), we must ensure the denoising-level action likelihoods $a_{k-1,t} \mid a_{k_t}, s_t$ are non-singular. ReinFlow [Zhang et al., 2025] modifies the bi-level PPO algorithm of [Ren et al., 2024] for flow-based policies, achieving this by adding additional Gaussian noise to each flow step. For given choice of noise levels σ_k^2, this yields the following inference procedure:

$$a_{k-1} \sim \bar{\pi}^{\text{FLOW}}(\cdot \mid a_k, k, s) := \mathrm{N}(v_\theta(a_k, \tfrac{k}{K}, s), \sigma_k^2 \mathbf{I}) \tag{3.4}$$

In OGPO, we anecdotally observe that naively adding noise can degrade policy performance by changing the marginal distributions of actions $a_{t,k}$ generated during denoising. We therefore introduce a correction proposed by Albergo et al. [2023] which (in the infinite step limit) ensures the per-denoising-step marginal distributions of noise-augmented actions match those of standard flow sampling; see also Liu et al. [2025]. See Appendix E.2 for details.

Algorithm 1 OGPO (Abbreviated)

for each environment step until `done` **do**
  Execute $a_t \sim \pi_{\bar{\theta}}(\cdot \mid s_t)$, and update buffer $\mathscr{B} \leftarrow (s_t, a_t, r_t, s_{t+1}, \texttt{done})$.
  `% Standard Critic Update`
  Update critic networks $\phi_1, \ldots, \phi_M$ using empirical TD Error (2.3) over $\mathscr{B} \sim \mathscr{D}_{\text{roll}}$.
  `% Actor Update via Multiple Denoising Trajectories`
  for $i = 1, \ldots, N_{\texttt{batch}}$ **do**
    Sample state $s^{(i)}$ from $\mathscr{B}$, and action trajectories $a_{K:0}^{(i,j)} \sim \pi_{\bar{\theta}}(\cdot \mid s^{(i)})$ for $1 \le j \le N_{\texttt{group}}$.
    Estimate value baselines via $\hat{V}^{(i)} \leftarrow \frac{1}{N_{\texttt{group}}} \sum_j Q_{\text{targ}}(s^{(i)}, a_0^{(i,j)})$
  end for
  Update actor using aggregated PPO loss (3.3)
  `% EMA parameters`
  Update target parameters $\bar{\theta} \leftarrow (1-\tau)\bar{\theta} + \tau\theta$, $\bar{\phi}_i \leftarrow (1-\tau)\bar{\phi}_i + \tau\phi_i$. Set $Q_{\text{targ}} = \frac{1}{M}\sum_i Q_{\bar{\phi}_i}$.
end for

OGPO: Core Insights

- **Off-Policy Q-learning for On-Policy GCP Extraction.** OGPO severs the bi-level MDP at environment transitions, using Q_{targ} as a terminal reward for the GCP's denoising MDP.
- **0^{th}-order optimization.** PPO-style zeroth order optimization over denoising trajectories avoids first order backpropagation through Q_{targ} and the GCP chain. This simplifies RL-finetuning in high-Lipschitz tasks.
- **Debiased noise injection.** SDE-augmented flow steps yield non-singular likelihoods for the PPO ratio ω_θ, with the stochastic interpolant correction ensuring marginal distributions match standard ODE inference. All $N_{\texttt{batch}} \times N_{\texttt{group}}$ trajectories are sampled and scored in parallel.

4 Improving OGPO by Mitigating Critic Over-exploitation

In this section, we identify a major limitation of OGPO: over-exploitation of learned critics due to highly expressive policy updates (Section 4.1) . We then introduce two modular modifications to OGPO which overcomes this tendency, and which are mutually compatible:

- OGPO+ (Section 4.2), which combines OGPO with behavior cloning regularization on *success-only* trajectories

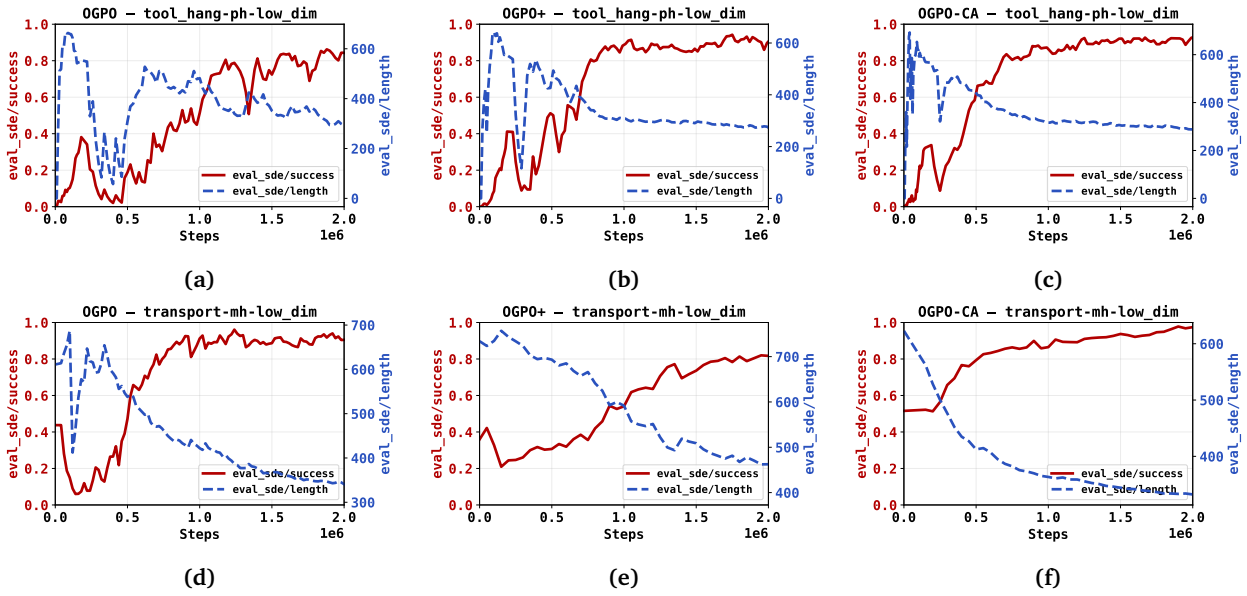


Figure 4: The above plots show the full training comparison between (a) Vanilla **OGPO**, (b) **OGPO+**, and (c) **OGPO+CA**, on state-based ROBOMIMIC tasks. The red axis shows success rate and the blue axis shows the mean length of *successful* trajectories.By aggressively maximizing sparse reward, **OGPO** optimizes for both task success rate, and *completion in few steps*. Without further regularization, the two can be in tension, causing a sharp initial decrease in the length of the policy rollouts, subsequent oscillations in success rates (TOOLHANG, (a) a high-precision task) or plateaus in performance (TRANSPORT, (d) a long-horizon task) . By adding a success buffer, **OGPO+** bias the policy learning objective to favor task success (b, e)). **OGPO+CA** mitigates the effect of outliers in the critic estimation, thereby fitting the "dip" between offline BC and online training (c, f).

- **OGPO+CA** (Section 4.3), which uses a *conservative advantage* for policy extraction, thereby *drastically mitigating the "dip" in offline-to-online adaptation*

For a practitioner, we recommend using **OGPO+CA** for highly stable policy extraction. In addition to these modifications, we optionally incorporate Q-variance reduction that averages the TD targets over N_{vr} actions sampled from the reference actor, thereby improve critic accuracy:

$$L_{\text{critic,vr}}(\phi) = \mathbb{E}\Big[Q_\phi(s_t, a_t) - \Big(r_t + \gamma \cdot \frac{1}{N_{\text{vr}}} \sum_{i=1}^{N_{\text{vr}}} Q_{\text{targ}}(s_{t+1}, a_{t+1}^{(i)})\Big)\Big]^2, \quad a_{t+1}^{(i)} \overset{\text{i.i.d}}{\sim} \pi_{\tilde{\theta}}(\cdot \mid s_{t+1}). \tag{4.1}$$

We find that Eq. (4.1) always yields some improvement, but this benefit is most pronounced in pixel-based environments ; it makes a marginal difference in state-based runs on pre-training from full data.

Experimental Setup: To identify improved design choices, we consider experiments on the state-based and image-based ROBOMIMIC tasks, which are described in greater detail in Section 5.1. For state-based runs, we use all state-information directly; for image based runs, we pass image observations using a frozen PaliGemma VLM backbone from the pre-trained π0.5 VLA [Intelligence et al., 2025]. Further details are given in Appendix I.

4.1 Vanilla OGPO Over-exploits Imperfectly Learned Critics

Recall that **OGPO** makes PPO-style updates to the denoising MDP. The combination of the expressive generative policies and PPO updates on the full-denoising trajectory risks causing OGPO to over-optimize the critic, overfitting to advantages which are poorly estimated.

Success-Speed Tradeoff. The typical "sparse-reward" manipulation setting assigns reward of -1 each time step a task remains uncompleted. Thus, minimizing cumulative reward introduces a tension between completion *rate* and completion *speed*. As a result, OGPO may attempt to finish tasks too quickly, causing success rates to drop, harming future exploration training stability. This success-speed tradeoff is visible in Figure 4a, where we see average task length rapidly improves, but success rate plateaus. Anecdotally, we found that the variance-reduced critic update Eq. (4.1) did not improve this tradeoff.

Overexploitation is Exacerbated in Pixel-Based RL. We observe that vanilla OGPO has more severe exploitation in pixel-based settings. We consider a ROBOMIMIC SQUARE environment described above, where pixels are feautrized using a frozen PaliGemma VLM backbone from $\pi 0.5$. To isolate the effects of pixels, we compare four variants: (1) state-based actor/state-based critic; (2) pixel-based actor/state-based critic (3) pixel-based actor/pixel-based (4) state-based actor/pixel-based critic. We plot variants (1-3) in Figure 5, and omit (4) due to collapsing runs. We that the policy trains effectively for both state-based critic runs (1)&(2), but fails on (3)&(4), suggesting that *pixel-based* critics prevent learning. We hypothesize that such critics learn less accurately due to the richer observation space, making them more susceptible to exploitation via OGPO. Anecdotally, we found that the variance-reduced critic update made modest but very limited improvements to the pixel-based critics, suggesting the need for further interventions.

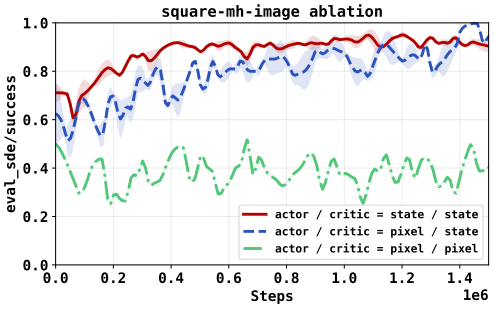


Figure 5: On ablating actor and critic observation modalities, we observe that vanilla OGPO fails to improve policy performance from image-based critics.

4.2 OGPO+: Regularizing OGPO With Behavior Cloning of Successful Trajectories

To remedy critic overexploitation, OGPO+ incorporates a regularization term applied only to actions from successful trajectories. This biases policy improvement toward replicating only the actions that led to success [Oh et al., 2018]. Specifically, we maintain a *success buffer* $\mathscr{D}_{\text{succ}} \subseteq \mathscr{D}_{\text{roll}}$ containing transitions from episodes that achieve task success. During training, we sample mini-batches from $\mathscr{D}_{\text{succ}}$ and compute

$$L_{\text{BC}}(\theta) = \mathbb{E}_{(s_t^{\text{succ}}, a_{t,0}^{\text{succ}}) \sim \mathscr{D}_{\text{succ}}} \left[\text{BCLOSS}(\bar{\pi}_\theta(\cdot \mid s_t^{\text{succ}}), a_{t,0}^{\text{succ}}) \right] \tag{4.2}$$

where BCLOSS is the appropriate behavior cloning objective (e.g., denoising score matching for diffusion policies, or flow matching loss for flow policies). Success-imitations ground the policy toward known good actions, while the PPO objective more aggressively explores improvements. For OGPO+, the total policy loss combines both terms:

$$L_{\texttt{Total}}(\theta) = L_{\texttt{PPO}}(\theta) + \lambda_{\text{BC}} L_{\text{BC}}(\theta). \tag{4.3}$$

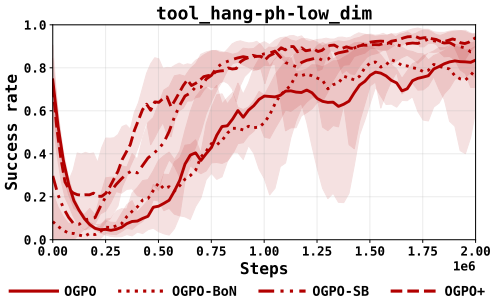


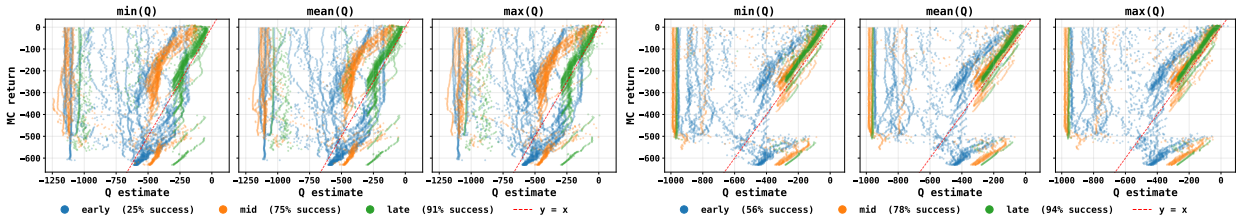
Figure 6: We perform a small sweep of ablations adding Best-of-N (BoN) Inference and Success Buffer on ROBOMIMICTOOLHANG.

(Optional) Best-of-N Inference. In many domains, such as language modeling, evaluating the quality of an action, or "verification" is learned more quickly and accurately than "generation" of good actions. This verification-generation gap [Setlur et al., 2025] motivates the popular practice of Best-of-N sampling [Brown et al., 2024], where one generates multiple proposal actions, and selects the best using a learned verifier.

Best-of-N sampling has seen widespread adoption in RL training of robotics policies [Mark et al., 2024, Dong et al., 2025, Li et al., 2025], using the target critic as verifier. In, **OGPO+** we do the same with a slightly modified critic Q_{BoN} described in Appendix A.1. We remark that, due to the aggressive policy extraction, Best-of-N inference yields only **marginal additional** performance; the success buffer, as described above, is crucial. Thus, we recommend *omitting* Best-of-N when inference cost is constrained.

$$a_{\text{BoN},t} := \arg\max\{Q_{\text{targ}}(s_t, a_{t,0}^{(i)}) : a_{t,0}^{(1)}, \ldots, a_{t,0}^{(N)} \overset{\text{i.i.d}}{\sim} \pi_{\theta_{\text{EMA}}}(\cdot \mid s_t)\}. \tag{4.4}$$

4.3 OGPO+CA: Mitigating the Offline-to-Online Performance Dip via Conservative Advantages



(a) OGPO **(b)** OGPO+CA

Figure 7: We take early-, mid-, and late- training checkpoints for **OGPO** and **OGPO+CA** to rollout 32 trajectories and visualize the min, mean, and max Q vs ground-truth, Monte-Carlo returns. (a) Shows **OGPO**'s Q values fluctuating widely between over- and under-estimating returns. (b) Shows **OGPO+CA**'s Q values converging more stably around the $y = x$ axis, demonstrating Q values accurately estimating returns.

A second challenge in offline-to-online RL is the pervasive "dip" in performance that arises transitioning from offline pretraining to online RL. Warm-starting methods like [Uchendu et al., 2023, Zhou et al., 2024] propose the use of high update-to-data (UTD) ratios and/or offline datasets during online RL, and the use of pessimistic critic updates. Anecdotally, we find that neither of these methods suffice. Moreover, from Figure 7a, we see that both over- and under-estimation of the Q values are possible, and both outliers potentially destabilize training. Thus, we instead to have the policy extraction step maximize the **conservative advantages**. This is made possible because our zero-order extraction takes advantages directly, and also accounts for the fact that global additive errors in critic values are less salient than incorrect *advantage* estimation.

For a given action a_i, we set

$$\hat{A}_j^{\text{cons}} = \begin{cases} \min_m A_{j,m} & \text{if } \min_m A_{j,m} > 0, \\ \max_m A_{j,m} & \text{if } \max_m A_{j,m} < 0, , \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

where we recall $A_{j,m} = Q_{\phi_m}(s^{(i)}, a_0^{(i,j)}) - \frac{1}{N_{\text{group}}} \sum_{i'=1}^{N_{\text{group}}} Q_{\phi_m}(s^{(i)}, a_0^{(i,j)})$ is the group-wise advantage using the m-th network in the ensemble. Eq. (4.5) provides a non-zero advantage (and thus updates the policy) if and only if *all advantages* have the same sign, thereby robustifying updates to estimation errors in the critic networks. As shown in Figure 7b, we we see that policy extraction with conservative advantages also improves the calibration of critic estimation, in that critic values in earlier states of training more tightly track those in later stages.

4.4 Conservative Advantages (OGPO+CA) Enable Stable Training on Images

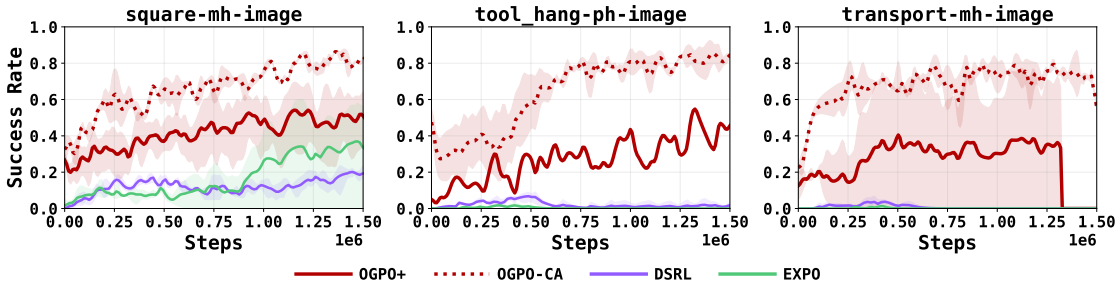


Figure 8: We compare **OGPO+** and **OGPO+CA** on ROBOMIMIC tasks with image-observations

Finally, we consider ROBOMIMIC tasks with image observations paired with robot proprioception information as a challenging setting for Q-learning and subsequently, policy extraction. From Fig. 8, we see that merely SFT via success buffer is not sufficient to guide policies to convergence. We observe that besides preventing the offline-to-online performance "dip", **OGPO+CA** also plays a crucial role in stabilizing policy improvement in high dimensional settings where learning Q-values over the large embedding spaces, proprioceptions, and actions is challenging. Moreover, baselines such as **DSRL** and **EXPO** fail to converge in image-based settings with no offline data in the replay buffer.

5 When does Full-Finetuning (OGPO) Improve Over Popular Baselines?

In this section, we carefully compare **OGPO**, **OGPO+**, and **OGPO+CA** to a number of popular baselines to elucidate the merits and limits of its design philosophy— full policy fine-tuning, off-policy critic learning, and PPO policy extraction. Our experiment environments are representative of many common challenges in robot learning (e.g. high precision, long horizon, mixed data quality), and baselines cover competing design philosophies (e.g. steering, residual learning).

Criterion	OGPO	QC	DSRL	EXPO
Mixed Data Quality	✓✓	✓✓	✓✓	✗✗
High Precision Tasks	✓✓	~~✓~~✓	✗✗	✓✓
Partial Demonstrations	✓✓	✓✓	~~✓~~✓	✗✗
Long Horizon	✓✓	✓✗	✓✗	✗✗
Dense/Dexterous	~~✓~~✓	~~✓~~✓	✓✗	✓✓
High Sample Efficiency	✓✓	~~✓~~✗	✗✗	✓✗

Table 1: Left (resp. right) symbol indicates achieving high success With (resp. Without) task-specific hyperparameter tuning. ✗- fails to converge on all tasks; ✓- converges on some but not all tasks; ~~✓~~- converges on all tasks, but below SOTA success/efficiency; ✓- converges on all tasks, competitive with SOTA success/efficiency. We use the optimized variants where possible (e.g. **OGPO+** for **OGPO** and similarly for all the baselines).

Summary of Findings. We summarize comparisons to other off-policy methods in Table 1. Each method has two columns: left denotes if the method converges with task-optimized hyperparameters, and right denotes fixed hyperparameters across all tasks within the criterion (see Appendix J). The markings are explained in the table caption.

We find that **OGPO is able to learn in sparse-reward tasks with mixed/partial data quality and on high-precision/long horizon tasks**, whereas other methods struggle in one or more of these regimes. It also exhibits (often times drastic) gains in sample efficiency compared to these methods, and order-of-magnitude improvements related to the on-policy DPPO algorithm. However, **OGPO** is less performant on the dense-reward tasks from the Adroit Hand benchmark (Figure 11).

Comparisons are detailed further in Section 5.2. Sample efficiency improvements v.s. DPPO are expected (off- vs. on-policy), and we attribute gains against off-policy baselines to exploration behavior and expressive policy updates, studied in Section 6.1. Appendix H ablates the merits of zero-order policy updates vs. backpropagation through time, the role of *negative-advantage gradients* in encouraging exploration, and the enhancements distinguishing OGPO and OGPO+.

5.1 Experimental Setup

Baselines. We compare against the baselines mentioned in Section 7.3, which are described in more detail in Appendix F In short, we consider: (i) DPPO [Ren et al., 2024], representative of on-policy learning, (ii) DSRL [Wagenmaker et al., 2025], representative of off-policy noise steering (iii) EXPO [Dong et al., 2025], representative of learning residual corrections to the GCP, and (iv) to a variant of QC [Li et al., 2025] representative of behavior cloning policy extraction. We do not compare to ReinFlow [Zhang et al., 2025] due to reported reduced sample efficiency compared to DPPO, making the latter a more compelling baseline. We also skip comparison to PA-RL [Mark et al., 2024] for reasons described in Appendix F.6. Lastly, we introduce a steering+residual learning baseline, (v) S/R, combining DSRL and EXPO to (hypothetically) yield the benefits of both. For a fair comparison with OGPO+, we implement each baseline with its own best-practices, as described in Appendix F.

Environments. Our simulation environments are chosen to elicit key challenges faced in modern robot learning: *Robomimic:* To test high-precision robotic control, we use three ROBOMIMIC tasks [Mandlekar et al., 2021]: SQUARE (medium-horizon insertion), TOOLHANG (long-horizon multi-step insertion), TRANSPORT (bi-manual long-horizon transfer). SQUARE and TRANSPORT use Multi-Human (MH) datasets; TOOLHANG uses Proficient-Human (PH) with BC stopped at 50% success. *Franka Kitchen:* We use the FRANKA-KITCHEN benchmark [Gupta et al., 2019] with a Franka robot manipulating 4 kitchen objects, testing sensitivity to multi-step trajectories with complete demonstrations (KITCHEN-COMPLETE), randomized subtask orders (KITCHEN-MIXED), and sequential partial trajectory data (KITCHEN-PARTIAL). *Adroit:* To test performance in dextrous manipulation tasks with dense-reward, we use the 24-DoF Adroit Hand benchmark:DOOR-V1, HAMMER-V1, PEN-V1, RELOCATE-V1 for door opening, hammering, pen reorientation, and object relocation. Expert datasets from D4RL/Minari. *LIBERO:* Finally, to test image-based language-conditioned manipulation, we use the ROBOMIMIC and LIBERO benchmarks [Liu et al., 2023]. Further details are given in Appendix I.

Experimental Regime: Online RL from a BC Checkpoint. We emulate real-world robot learning settings where large-scale pretrained policies with varying levels of online success rates are deployed to learn novel tasks **without access to offline datasets during online RL**. Thus, we pre-train a flow GCP for all baselines, clip it to at most 50% success rate, and use the same BC checkpoint for all baselines in online RL without additional data. Full details in Appendix J.1.

5.2 Comparison to other methods

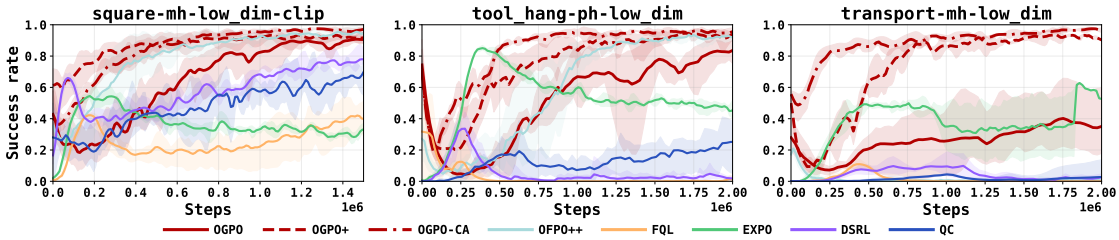


Figure 9: Comparison with natural off-policy baselines (EXPO, DSRL, QC), and on-policy algorithms modified to use OGPO-style off-policy value functions (OFPO++, FQL) on ROBOMIMIC SQUARE, TOOLHANG, and TRANSPORT.

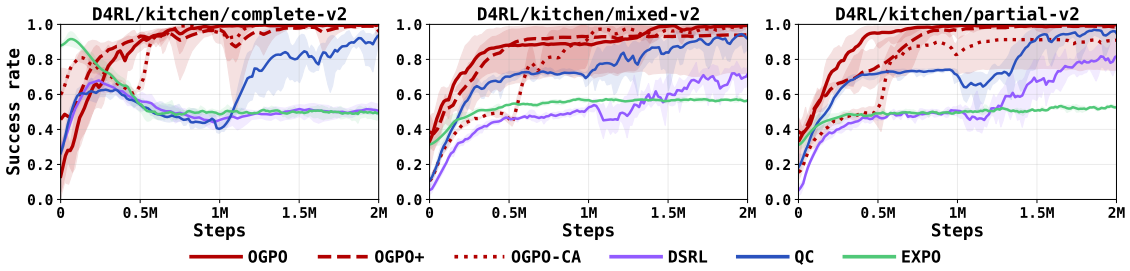


Figure 10: Comparison against natural off-policy baselines (EXPO, DSRL, QC) on FRANKA-KITCHEN

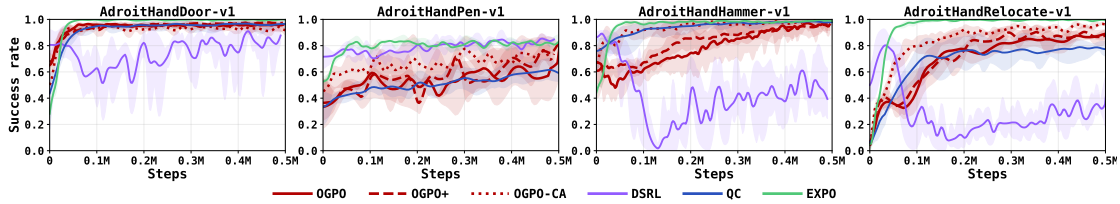


Figure 11: Comparison against natural off-policy baselines (EXPO, DSRL, QC) on the `AdroitHand`

Expressivity: Full-Policy Finetuning (OGPO) vs. Steering (DSRL) vs. Residual (EXPO). Next, we compare **OGPO** to performant off-policy alternatives that do not fine-tune the full GCP across 11 aforementioned tasks. *Steering* (**DSRL**) can be sample-efficient but relies on sufficient base policy action coverage, leading to suboptimal performance when the base policy's performance is poor, such as in KITCHEN tasks. Further, by not updating later steps of the GCP, steering struggles on high-precision tasks such as the ADROIT task suite. We also empirically found it to be sensitive to hyperparameters; in some tasks, DSRL performance crashes despite heavy tuning. We attribute some of this instability to our use of **DSRL** on a flow-based GCP instead of a diffusion-based GCP; the original paper uses diffusion GCPs for low-data-coverage experiments. However, we are also more sample-efficient than **DSRL**'s paper-reported numbers on shared tasks.

Residual learning (**EXPO**) performs well when the base policy is strong and thus only minor residual corrections are needed (it is highly performant in ADROIT in Figure 9), but, like steering (**DSRL**), it generally performs poorly or is unstable when the base policy performance starts lower (KITCHEN and most ROBOMIMIC tasks). We note that when given offline data, **EXPO** can perform well (see Figure 11), but our experimental regime is without access to the pre-training data. Our *Steering + Residual Learning* (**S/R**) baseline combines **EXPO** and **DSRL**; we plot sample efficiency curves in ROBOMIMIC tasks in Figure 24, where we see that it is better than **EXPO**/**DSRL** alone in SQUARE, albeit still worse compared to **OGPO**, and demonstrates unstable training in the high precision TOOLHANG task.

Off-Policy Learning vs. Self-Distillation/Behavior Cloning (BC) with QC. Next, we compare *policy extraction* methods. We find the action-chunked **BPTT** variant proposed in the **QC** paper to perform poorly (Fig. 16) on flow policies, and thus use a variant that explores online with Best-of-N action sampling and fine-tunes the BC policy on transitions from the online replay buffer. **QC** plateaus at lower performance for most tasks, requires more task-specific hyperparameter tuning, and has worse sample efficiency. We attribute this to SFT's inability to expand the support of the GCP action distribution, required for sufficient exploration.

Off-Policy OGPO vs. On-Policy DPPO. Finally, we compare **OGPO+** against **DPPO**, where the major difference between the two is that **OGPO+** truncates the bi-level MDP proposed by **DPPO** at the end of each denoising trajectory with terminal rewards coming from an off-policy Q-function,

while DPPO treats the entire bi-level MDP as a single MDP to train with on-policy RL. On final success rates across ROBOMIMIC SQUARE and TRANSPORT, this off-policy modification results in DPPO taking $\sim 10\times$ longer to reach the final success rates achieved by OGPO+. Overall, we find that both OGPO and OGPO+ outperform DPPO's paper-reported results in both sample efficiency and final performance across all shared tasks, even with matched network architectures and action chunk lengths.

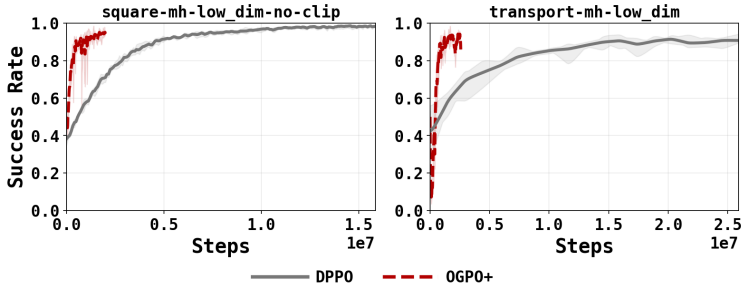


Figure 12: OGPO+ substantially improves sample efficiency compared to the on-policy DPPO algorithm.

Summary: OGPO outperforms natural baselines

OGPO outperforms all natural off-policy baselines in sparse reward precise manipulation settings, and is an order of magnitude sample efficient than on-policy methods with minimal hyperparameter tuning.

6 Understanding and Ablating The Merits of OGPO

6.1 Does OGPO Encourage Exploration?

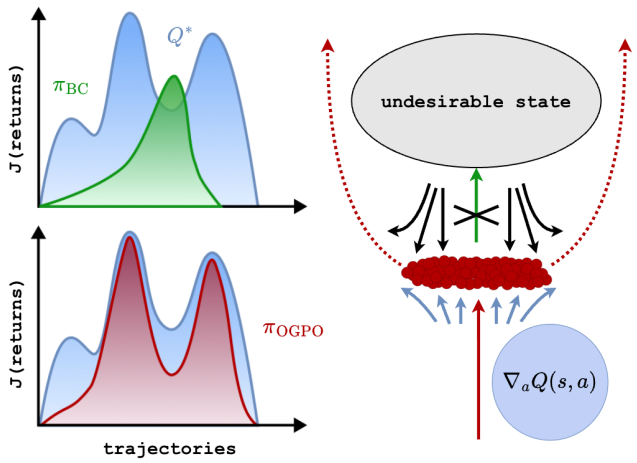


Figure 13: Left: Consider a policy with two equally near-optimal modes that are only weakly covered by the BC data (green). OGPO maintains coverage of both modes even after convergence. *How?* **Right:** We illustrate our mental model with an example where bi-modality arises from a bifurcation around an obstacle or undesirable state, shown in gray. In this setting, $\nabla_a Q(s,a)$ points toward the obstacle, while directions orthogonal to $\nabla_a Q(s,a)$ move perpendicular to it. By preserving action variance orthogonal to $\nabla_a Q(s,a)$, OGPO maintains coverage over action chunks that can separate into the "left" and "right" trajectory modes.

By aggressively exploiting the critic (Section 4), OGPO generates actions beyond the support of the offline data distribution used in the BC phase (Figure 13, *left*), resulting in high task success as well high *task efficiency*, measured in terms of time-steps to completion. Here, we identify a surprising finding:

> OGPO **generates highly diverse trajectories**, despite aggressively exploiting the critic for high success rates and task efficiency.

Whereas diversity, optimality and task efficiency are often regarded as being at odds [Huang et al., 2025a, Setlur et al., 2025], we show that OGPO accomplishes all simultaneously. Below, we present extensive evidence for this finding, and propose a mental model, summarized in Figure 13, *right*, as to how OGPO achieves this affect.

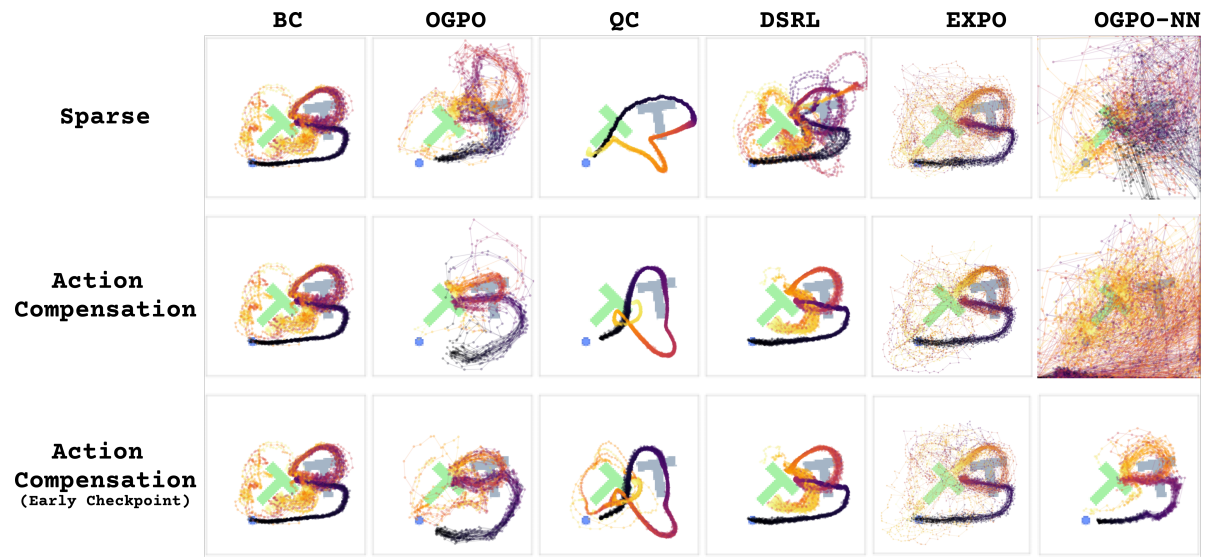


Figure 14: We plot 50 rollout trajectories on the `pushT` task with (top) sparse reward, (middle) sparse reward with Δa_t compensation, and (bottom) early-stage sparse reward with Δa_t compensation policy. Compared to the baselines, we observe **OGPO** learns policies with faster execution, minimal fine adjustments, and action spaces wider than the BC distribution in the sparse reward setting. The observation that action compensation forces OGPO to adhere to the rollouts in the vicinity of BC policies further confirms **OGPO**'s exploration tendencies.

OGPO drives greater trajectory diversity. We study the `PushT` task [Chi et al., 2023], a classical example of trajectory-level multimodality, where a blue-dot pushes a gray "T" to the green goal configuration (Figure 14). We consider two reward settings: the classical sparse reward $r = -\mathbf{I}\{\text{not done}\}$, and an "action-compensated" reward $r = -(\mathbf{I}\{\text{not done}\} + \lambda\|\Delta a_t\|)$ which penalize per-step action magnitudes (due to PushT's physics enabling unbounded actions). We compare **OGPO** against natural baseslines and visualize the learned trajectories in Figure 14. Here, dark points represent the initial actions in the trajectory, and the color lightens to yellow ones as the time-step progresses. In the absence of action-compensation, **OGPO** learns to take larger action that complete the trajectory in fewer time steps (task-efficiency), and with full success.[3]

Still, **OGPO** *preserves a relatively wide manifold of valid actions* [Ren et al., 2024], and seems to *preserve additional trajectory-level modes*. On adding an action compensation term, **OGPO** takes smaller steps and prunes many of its modes, favoring modes which allow shorter path-length. This makes sense as **OGPO** directly exploits the critic, yielding actions closer to optimal and further from the base policy.

Comparing to the baselines, **QC** and **DSRL** show limited manifold expansion, remaining closer to the BC initialization. **EXPO**'s residual policy facilitates support expansion but not optimal policy extraction. This can be seen by a range of corrective actions being taken near the T-shape handle. Lastly, we test **OGPO–NN**, which zeros out all negative advantages and retains only positive advantages. Whereas prior work [Setlur et al., 2025] would suggest that negative advantages *increase exploration*, we find that they also seem necessary for "sharpening" [Huang et al., 2025a] towards optimal modes.

OGPO preserves action variance "orthogonal" to task success. We now uncover the *concrete mechanism* by which OGPO preserves both *action* and *trajectory*-level diversity. We compare **OGPO** to relevant baselines on TOOLHANG, isolating two critical states at times $t = 9$ (the needle being transported toward the hole) and $t = 28$ (the wrench being inserted) from a single, shared demonstration trajectory with maximal variance in Q-values. Each policy is trained from the same BC checkpoint, removing spurious

[3]Note that without action-compensation, path-*length* is not constrained, only time to completion.

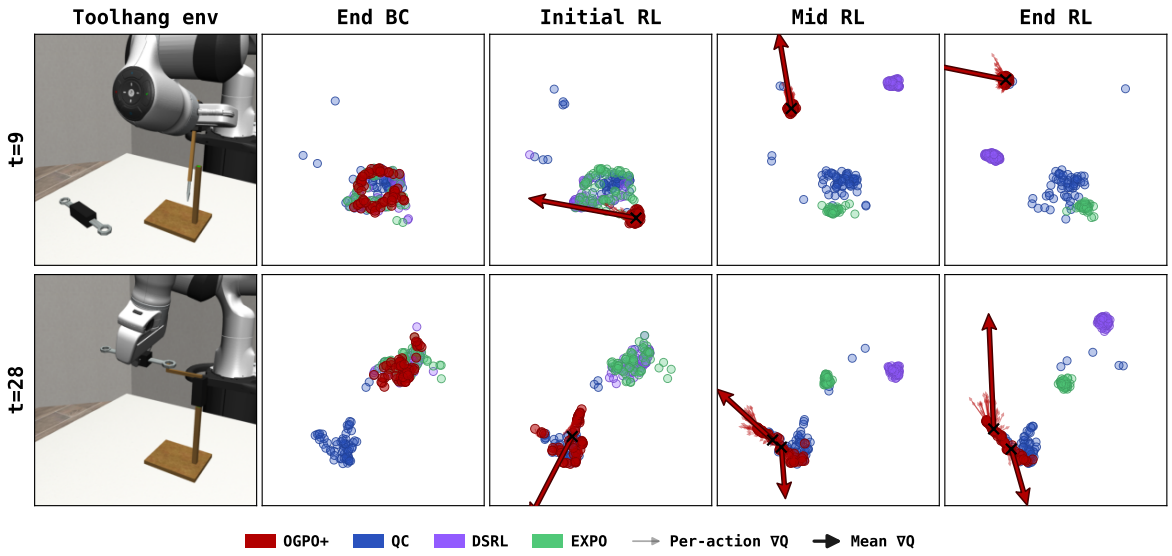


Figure 15: We plot the UMAP embeddings [McInnes et al., 2018] of actions generated via **OGPO+** and the natural baselines. We show the Q-function gradients with respect to **OGPO+** actions ($\nabla_a Q(s,a)$) projected in the same UMAP space as well as a vector sum of the per-action $\nabla_a Q(s,a)$ denoted as the consensus ∇Q. **OGPO** actions show a sharp variance reduction compared to the baselines, especially in axes orthogonal to the consensus ∇Q direction.

variation[4].

For each time step, we pool together 64 actions from policies trained with each baseline, compute a common UMAP embedding [McInnes et al., 2018], and visualize them in Figure 15, color coding actions by method. For the **OGPO** actions, we also plot arrows that compute the gradient $\nabla_a Q(s,a)$ of the mean Q function (from after training), which gives the local direction of steepest ascent for actions to improve the critic value (see caption for details). To visualize the $\nabla_a Q(s,a)$ for each action a_i, we compute unit vectors $u_i = \frac{\nabla_{a_i} Q(s,a_i)}{||\nabla_{a_i} Q(s,a_i)||}$ and measure an agreement score $\psi = ||\frac{1}{N}\sum_i^N u_i||$. When $\psi > 0.6$ we consider majority of actions having the same $\nabla_a Q(s,a)$ unit vectors, and $\psi \leq 0.6$ as there not being a consensus, at which, we compute K-means clusters over $\nabla_a Q(s,a)$ with cluster centers shown as black crosses in Figure 15. We include snapshots across four phases of training, from offline to completion.

Our findings reveal **that OGPO increases variance in a *selective* manner.** At $t = 9$, there is minimal trajectory level diversity due to the ensuing precision requirements. Thus we see **OGPO** exhibits the *most aggressive* shrinking of action variance. However, at $t = 28$, greater action action variance is permitted, and preserved even at the *end of training* (Figure 15, *bottom right*). However, **OGPO** does not increase variation isotropically: rather, the remaining action-variance as even *orthogonal* to the critic gradient. Note that, along these directions, differences in actions have *zero effect on critic values*, to first order. Therefore, we find the **OGPO** *allocates large variance along directions which do not affect task success*. At the same time, **OGPO** (a) sharpens the distribution orthogonal to these directions (resulting in the "thin" ellipsoid seen in Mid/End training in at $t = 28$), while (b) aggressively "stretching" the action distribution to align with critic gradients in parts of the action distribution when gradients $\nabla_a Q(s,a)$ exhibit strong consensus, e.g. $\psi > 0.6$. Thus, **OGPO** can *both* optimize the critic for task performance/completion time while *simultaneously* preserving as much action diversity as possible.

[4]note that **QC** uses additional critic distillation during the BC phase, leading to different actions after offline training

Mental model: mode-preservation via orthogonal action variance. Here, we propose a mental model for how OGPO's selective "stretching" tendency preserves trajectory-level multimodality. We depict a mental-model of this trajectory-level multimodality in Figure 13. We consider a task with two modes—navigating left or right around an obstacle. The optimal branching point is depicted with (*red dots*). Below these, the optimal actions approach the obstacle, whereas above, they move around it. In the center, $\nabla_a Q(s,a)$ points vertically (either up or down). OGPO preserves variance orthogonal to this direction, preserving actions which ultimately branch into the left and right-modes. Thus, allocating variance at the "decision point", while "stretching" actions at the extremes, sharpens the trajectory distribution around *both* feasible modes.

How does OGPO preserve exploration?

- For critical states precursor to future high precision demanding ones, OGPO learns a strong $\nabla_a Q(s,a)$ consensus and subsequently contracts the policy toward a narrow high-value action manifold.
- In states admissible of multiple near-optimal actions, OGPO preserves variance along directions approximately orthogonal to $\nabla_a Q(s,a)$, where action perturbations have negligible first-order effects on the value.

Why does OGPO preserve exploration? An important question to ask is: why does OGPO preserve exploration better than alternatives? A comprehensive account would warrant further study, which we defer to future work. Here, we hypothesize that the key factor which allows OGPO to preserve variance comes from finetuning all the steps of the generative process; in Appendix Fig. 23, we observe that other full-finetuning methods (e.g. FPO++ [Yi et al., 2026]), also preserve variance, though to a slightly lesser degree. We hope to pursue the full breadth of this question in a future study.

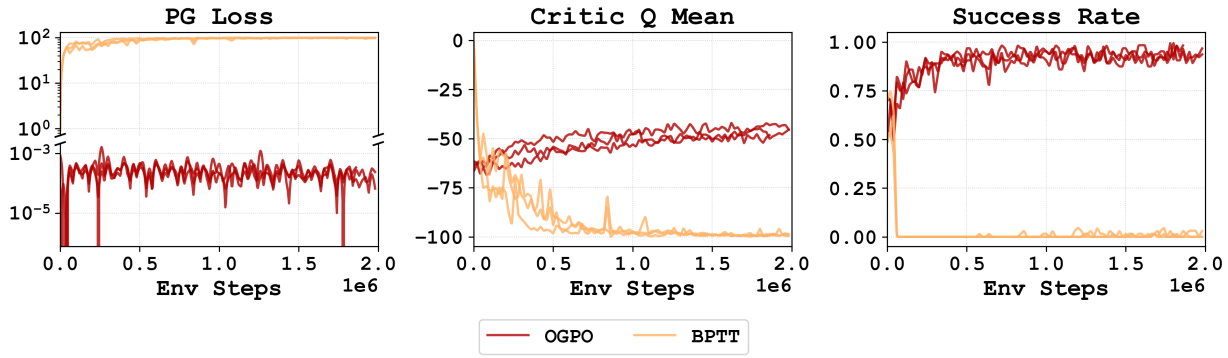


Figure 16: BPTT uses Q-values directly to backpropagate gradients along the entire GCP chain. This results in unstable gradients and poor convergence. In contrast, OGPO uses PPO-style policy gradient loss using Q-functions described Eq. (3.2). This results in stable gradients and sample-efficient convergence.

6.2 Does PPO policy extraction outperform natural alternatives (AWR, FPO)?

We observe that OGPO uses a simple API: apply *any RL* algorithm to the denoising MDP whose terminal rewards are given by the critic. Specifically, we can write a more general loss of the form:

$$\begin{aligned}\text{PolicyLoss}(\theta \mid \mathrm{s}, \mathrm{a}^{\mathrm{K:0}}, \hat{\mathrm{A}}^{\mathrm{G}}, \theta) = \mathbf{I}\{\hat{A}^G \geq 0\} \cdot \text{loss}_{(+)}(\theta; a^{K:0}, s) \cdot \text{weight}_{(+)}(\hat{A}^G) \\ + \mathbf{I}\{\hat{A}^G < 0\} \cdot \text{loss}_{(-)}(\theta; a^{K:0}, s) \cdot \text{weight}_{(-)}(\hat{A}^G)\end{aligned} \tag{6.1}$$

Method	$\hat{A}^G \geq 0$		$\hat{A}^G < 0$	
	$\text{loss}_{(+)}$	$\text{weight}_{(+)}$	$\text{loss}_{(-)}$	$\text{weight}_{(-)}$
OGPO	PPO likelihood+clip	$\hat{A}^G$	PPO likelihood+clip	$\hat{A}^G$
OGPO-NN	PPO likelihood+clip	$\hat{A}^G$	✗	✗
AW-OGPO	PPO likelihood+clip	$\exp\left(\hat{A}^G/\beta\right)$	PPO likelihood+clip	$\exp\left(\hat{A}^G/\beta\right)$
AW-OGPO-NN	PPO likelihood+clip	$\exp\left(\hat{A}^G/\beta\right)$	✗	✗
AWR-FM	CFM loss	$\exp\left(\hat{A}^G/\beta\right)$	CFM loss	$\exp\left(\hat{A}^G/\beta\right)$
OFPO++	exp(CFM))+clip	$\hat{A}^G$	(exp(CFM))+clip	SPO weighting

Table 2: We present a tabular description of the differences in the policy extraction algorithms that are compatible with the **OGPO**'s policy extraction framework given advantages ($\hat{A}^G$). Eq. (6.1) succinctly describes the combination of $\text{loss}_{(+/-)}$ and $\text{weight}_{(+/-)}$ that contribute to the policy loss for the different methods. Above, PPO-likliehood corresponds to Eq. (3.2), clipping clips likelihoods as in Eq. (3.2), β is a temperature hyperparameter chosen per-task, and CFM indicates the conditional flow matching loss [Lipman et al., 2022]. Further details are given in Appendix H.4

where the policy loss under a parameter θ, for state s, denoising chain $a^{K:0}$, and advantage estimate $\hat{A}^G$ consists of a loss depending on $s, a^{K:0}$, and weighting depending on the advantage. To compare with alternatives, we decompose the loss into a separate terms depending on the advantage sign.

We now describe a number of alternatives based on this formulation Table 2. First, we compare to **AW-OGPO**, which uses exponentiated advantages as in Peng et al. [2019], but instead reweighs the **OGPO** likelihood ratio given in Eq. (3.2). For both **OGPO** and **AW-OGPO**, we also introduce a positive-only variants of **OGPO-NN** and **AW-OGPO-NN** which zero the loss/weighting when advantages are zero. In addition, we introduce **AWR-FM**, a natural baseline which up-weights the conditional flow-matching (CFM) loss rather than PPO likelihoods. Generally, this underperforms **AW-OGPO**, so we omit the no-negative advantage variant. For all AWR-style runs, we perform per-task hyperparameter tuning to determine an optimal temperature parameter β to ensure a steelman comparison. Finally, we compare to extraction via FPO++ [Yi et al., 2026], which applies a number of novel design decisions detailed in Appendix H.4. All methods use the same replay data, critic training, and group-wise advantages.

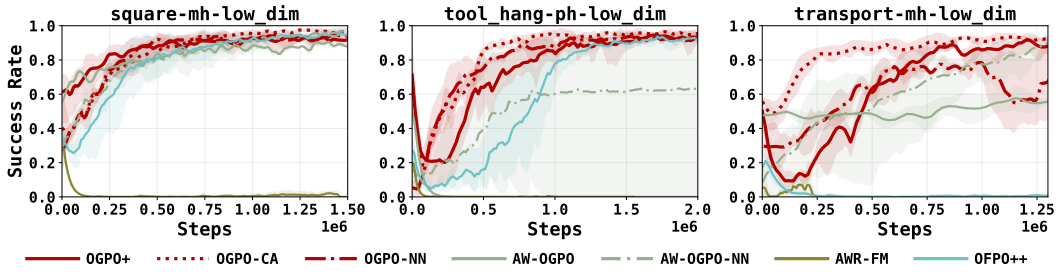


Figure 17: **OGPO** comparisons with policy extraction ablations with **AWR-FM**, **AW-OGPO**, **AW-OGPO** and **OFPO++** on ROBOMIMIC environments

As shown in Figure 17, **AWR-FM** fails across the ROBOMIMIC tasks, indicating that pure advantage-weighting of the flow loss is insufficiently expressive compared to likelihoods that use the full denoising MDP. Using the full likelihoods in **AW-OGPO**, positive-only **AW-OGPO**, and **OFPO++** yields stronger performance, although not on par with **OGPO+CA**. In particular, the positive-variant of **AW-OGPO** outperforms that of normal **AW-OGPO**, by virtue of being more aggressive (note that regular **AW-OGPO** still has positive weights on likelihoods when advantages are negative), but still cannot reach full success on TOOLHANG. On the other hand, **OFPO++** collapses on the long-horizon TRANSPORT task.

Unlike **AW-OGPO**/**AW-OGPO-NN**, removing negative-advantage gradients makes a minimal impact on **OGPO** for tasks like SQUARE and TOOLHANG, where merely imitating high-valued action samples is sufficient to sharpen policy distributions (Figure 17). However, for a task like TRANSPORT, where avoiding suboptimal policy modes is critical for task success, we observe worse performance for both **OGPO—NN** as well as **AW-OGPO-NN**. This suggests that negative advantages are important for mitigating suboptimal action distributions learned during pretraining.

Why PPO updates are optimal.

We find that PPO style updates provide the *most aggressive critic exploitation*, whereas AWR style advantages induce more modest updates that are suboptimal in online RL. Similarly, weighting denoising likelihoods (Eq. (3.2)) outperforms weighting the conditional flow-matching loss, again because the former is more aggressive. Stated succinctly, **just use the best *on-policy* RL algorithm, i.e. PPO, for extraction from the off-policy critic**!

OGPO enables consistent cross-task hyparameters

OGPO's non-exponentiated advantage weighting removes the β hyperparameter, which we find needs to be tuned for different tasks, due to sensitivity to advantage magnitudes. Thus, **OGPO** functions with the **same hyperparameters** across domains, making it more suitable to extensions for multi-task learning.

6.3 Which Further Design Decisions Explain the Performance of OGPO and OGPO+?

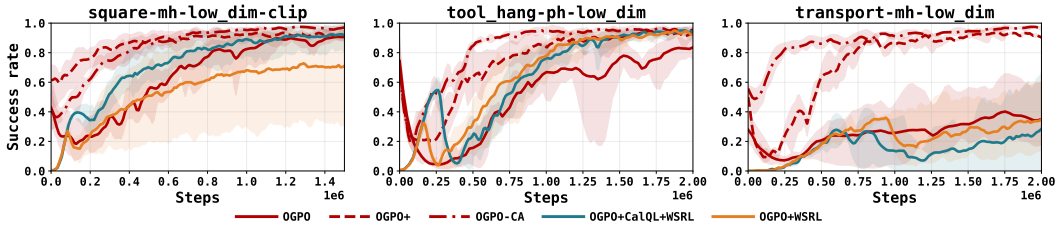


Figure 18: **OGPO+** and **OGPO+CA** obviate the need for offline-to-online Q-function RL

The following ablations are designed to systematically isolate various subcomponent decisions within **OGPO** and to explain which design choices align with maximizing sample efficient policy extraction. First, we compare **OGPO**'s zeroth-order policy extraction to Backpropagation Through Time (**BPTT**) that backpropagates first order gradients via Q functions and through the entire GCP denoising chain. As shown in Figure 16 directly backpropagating through the denoising chain often fails catastrophically, supporting our choice to optimize the GCP via importance sampling rather than through $\nabla_a Q(s,a)$.

Second, using Figure 19 as reference, Best-of-N inference provides only marginal gains by itself and can increase oscillations when the critic is imperfect. This is consistent with the role of Best-of-N as a verifier of critic learning at inference time, rather as a significant mechanism for policy improvement [Chow et al., 2025, Huang et al., 2025b]. In contrast, the success buffer used in **OGPO+** consistently improves sample efficiency and asymp-

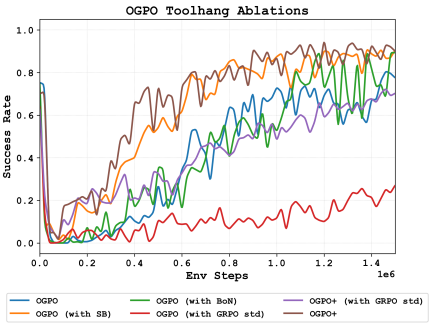


Figure 19: **OGPO** - **OGPO+** design ablations show that success buffer plays a crucial role in **OGPO+**'s performance, while Best-of-N plays the role of a verifier for improved critic learning by showing marginal improvements in performance.

totic performance by anchoring policy improvement to successful behavior. We provide a mathematical basis for the intuition that conditional flow matching (CFM) loss between $\bar{\pi}_\theta$, and the success buffer actions increases the GCP lower-bound on successful modes in Appendix E.3. Moreover, we modify the advantage computation from $\hat{A} = \frac{Q_{\text{targ}}(s_t, a_{t,0}) - \hat{V}}{\hat{\sigma}}$, where $\hat{\sigma}^{(i)} \leftarrow \sqrt{\frac{1}{N_{\text{group}}} \sum_j \left(Q_{\text{targ}}(s^{(i)}, a_0^{(i,j)}) - \hat{V}^{(i)} \right)^2}$ and find that GRPO-style variance normalization hurts performance. Finally, we ablate the offline-to-online Q-learning recipe proposed in Warm Start RL (WSRL, Zhou et al. [2024]) with and without Calibrated Q-Learning (CalQL, [Nakamoto et al., 2024b]), and compare against OGPO, OGPO+, and OGPO+CA. We find that CalQL+WSRL slightly improves vanilla **OGPO**, but fail to mitigate the policy collapse as prevented by **OGPO+** and **OGPO+CA**.

7 Related Work

We situate our work within the landscape of generative control policies, reinforcement learning for robotic control, and finetuning strategies for iterative generative models.

7.1 Generative Control Policies

The success of diffusion models in image generation [Ho et al., 2020, Song et al., 2020, Rombach et al., 2022] has inspired their adoption for robotic control. Diffusion Policy [Chi et al., 2023] demonstrated that denoising diffusion probabilistic models (DDPMs) can effectively parameterize visuomotor policies by iteratively denoising action sequences conditioned on observations. Flow-matching policies [Lipman et al., 2022, Liu et al., 2022] offer a more efficient alternative by learning velocity fields that transport noise to action distributions through ordinary differential equations (ODEs), achieving comparable performance with fewer integration steps.

Recent work has sought to improve the generative modeling capacity. Notably, shortcut models [Frans et al., 2024] condition on desired step sizes to enable few-step generation, while consistency models [Song et al., 2023] distill multi-step diffusion into single-step generation. Recently, [Pan et al., 2025] introduced Minimally Iterative Policies (MIP), demonstrating that two-step regression-based policies can match full flow model performance, suggesting that distributional learning may be less critical than previously believed. Orthogonally, tokenized autoregressive policies such as FAST [Pertsch et al., 2025] encode continuous action chunks via discrete cosine transforms to enable efficient training of vision-language-action (VLA) models on high-frequency control data.

For **OGPO**, we demonstrate flow and diffusion-based policies as representative of the general IGP formulation and leave generalization to other formulations as future work.

7.2 Reinforcement Learning for Robotic Policy Finetuning

The incorporation of Reinforcement Learning (RL) into robotic policy training mirrors the post-training paradigm in large language models [Ouyang et al., 2022, Shao et al., 2024]. On-policy methods such as REINFORCE [Williams, 1992] and PPO [Schulman et al., 2017] update policies using only data from the current policy iteration, ensuring stable but sample-inefficient learning. DPPO [Ren et al., 2024] extends PPO to diffusion policies by computing policy gradients through the denoising chain, while Reinflow [Zhang et al., 2025] applies similar principles to flow-matching policies.

Off-policy algorithms promise greater sample efficiency by maintaining replay buffers of past experiences. Classical approaches such as SAC [Haarnoja et al., 2018], TD3 [Fujimoto et al., 2018], and REDQ [Chen et al., 2021] learn Q-functions from off-policy data to guide policy updates. Temporal difference learning mitigates the requirement of the policy to compute Monte Carlo return to the go. However, naive application to IGPs in the RL-finetuning regime can exhibit training instabilities due

to large initial distributional shifts and value overestimation. To mitigate these, [Mark et al., 2024, Li et al., 2025] proposed using Q functions merely to rank stochastic policy actions and fine-tuning the policy using the Best-of-N actions. However, driving policy improvement via Q-function ranking can be inefficient as it requires exploration away from the mean values of the flow policy.

Concurrently, RL-100 [Lei et al., 2025] presents a comprehensive real-world RL framework built on diffusion policies, demonstrating deployment-grade success rates across eight manipulation tasks. RL-100 adopts the same bi-level MDP formulation and clipped PPO surrogate as DPPO, unifying imitation and reinforcement learning under a single objective across both offline and online stages, and additionally incorporates consistency distillation for high-frequency deployment. While RL-100 demonstrates impressive real-world reliability, its policy optimization remains fully on-policy, requiring iterative offline data expansion to achieve sample efficiency. **OGPO** instead decouples the bi-level MDP via off-policy critic learning, achieving comparable or superior sample efficiency in simulation without requiring multiple rounds of offline RL pre-training.

7.3 Finetuning Strategies for Generative Control Policies

Existing approaches to finetuning GCPs differ along the axis of *what* is optimized. Steering methods, exemplified by **DSRL** [Wagenmaker et al., 2025], optimize the distribution over initial noise a_K while freezing the pretrained denoising network. This constrains policy improvement within the support of the pretrained IGP distribution. Residual policy approaches such as **EXPO** [Dong et al., 2025] train an additional network π^{res} that modifies the final action $a_{\text{res}} = \pi^{\text{res}}(a_{t,0}, s_t)$, allowing mode shifts within the BC policy support but fails to facilitate discovery of new behaviors.

Policy-agnostic RL (PA-RL) [Mark et al., 2024] and Q-chunking (**QC**) [Li et al., 2025] employ Q-functions to rank behavior cloned policies with high-value actions or use $\nabla_a Q(s, a)$. Q-learning with Adjoint Matching (QAM) [Li and Levine, 2026] uses adjoint matching to convert the critic's action-gradient into a step-wise training objective for expressive flow or diffusion policies, avoiding direct backpropagation through the full denoising process. In the image generation domain, Flow-GRPO [Liu et al., 2025] concurrently applied GRPO [Shao et al., 2024] to flow matching models for text-to-image alignment, sharing with **OGPO** the ODE-to-SDE conversion for injecting stochasticity into deterministic flow policies and the use of group-relative advantage estimation over parallel denoising trajectories. However, Flow-GRPO operates in the on-policy, bandit-like setting: rewards are terminal (image-level), the "environment" is a single-step generation with no dynamics, and advantages are estimated via group normalization of final rewards rather than learned Q-functions.

In contrast, **OGPO** addresses the multi-step robotic control setting, where off-policy TD-learning is essential for sample efficiency across long environment horizons, and the two-level MDP structure enables reuse of costly environment transitions while performing on-policy updates purely within the denoising MDP. However, in addition to zero-order optimization via Q functions, **OGPO** performs SFT via Success Buffer actions for enhanced sample efficiency.

8 Conclusion and Limitations

We introduce **OGPO**, an approach that combines the best of on-policy and off-policy methods for finetuning generative control policies (GCPs) and enjoys high success rates and sample efficiency across numerous tasks. However, **OGPO** still has limitations, the most important being that the parallel denoising rollouts required to estimate Q-values can be prohibitively expensive for large VLA models due to the high inference costs. Future work focusing on Q-function learning fidelity can help ameliorate this limitation by reducing the number of parallel GCP rollouts.

TL;DR: Takeaways

- **Takeway #1: OGPO** provides a mechanism for scaling training compute given a limited interaction with the environment. At a coarse level, the GRPO sampling, parallel denoising tracks per-state, full-policy finetuning, and updates to every step of the denoising process can be viewed as axes along which compute is expended (Section 3). Our findings suggest that, even within the standard Actor-Critic template, simply **increasing training time computation** can improve sample efficiency drastically (Figure 9).
- **Takeaway #2:** While critic learning is widely believed to be bottleneck in online RL, our findings suggest that **better policy extraction alone** can yield substantial improvements in training stability and sample efficiency (Section 6.2).
- **Takeaway #3:** RL finetuning need not cause "mode collapse" or "distribution narrowing." Surprisingly, **full policy finetuning can increase action diversity** and enhance policy exploration, despite the fact that one is only trying to maximize reward (with no explicit entropy penalties) (Section 6.1). Understanding this phenomenon is an exciting direction for future work.
- **Takeaway #4: Zero-order policy policy optimization** can be incredibly effective given sufficient computation as it avoids unstable gradients through denoising steps or critics, improving performance on high-precision tasks. Further, the form of the likelihood ratios still provides useful gradient information, and can move policy mass away from the BC distribution (Figure 13).
- **Takeaway #5:** Full-finetuning of GCPs can lead to issues of critic overexploitation. However, the best remedy is not to slow down learning through hyperparameter adjustments, enforce pessimistic policy/critic updates or regularize entropy/distance to the base distribution. Instead, **targeted interventions, like imitating successful trajectories** (**OGPO+**) or **modifying the advantages** (**OGPO+CA**) are both reliable, preserve training efficiency, and ameliorate the need for task-specific hyperparameter tuning (Sections 4.2 and 4.3).
- **Takeaway #6:** While there are many options for fine-tuning a multi-step GCP (AWR, FPO, etc), **simple PPO is the most effective**, most stable, and requires the least amount of hyperparameter finetuning.

Acknowledgments

MN would like to thank Qiyang Li for helping with the initial implementation, and Zhiyuan Zhou, Seohong Park and Aviral Kumar for their informative discussions. This research used the Savio computational cluster resources provided by the Berkeley Research Computing program at UC Berkeley. MS would like to thank Aviral Kumar and Andrew Wagenmaker for useful discussions. SBP would like to thank Steven Man, Andrea Bajcsy, and Ken Nakamura for their insightful discussions. We acknowledge support from the Toyota Research Institute (TRI) University 2.0 program.

References

Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. doi: 10.1109/72.279181.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized ensembled double q-learning: Learning fast without a model. *arXiv preprint arXiv:2101.05982*, 2021.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

Yinlam Chow, Guy Tennenholtz, Izzeddin Gur, Vincent Zhuang, Bo Dai, Aviral Kumar, Rishabh Agarwal, Sridhar Thiagarajan, Craig Boutilier, and Aleksandra Faust. Inference-aware fine-tuning for best-of-n sampling in large language models. In *International Conference on Learning Representations*, volume 2025, pages 78936–78959, 2025.

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

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures, 2018. URL https://arxiv.org/abs/1802.01561.

Mikhail Figurnov, Shakir Mohamed, and Andriy Mnih. Implicit reparameterization gradients. *Advances in neural information processing systems*, 31, 2018.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. *arXiv preprint arXiv:2410.12557*, 2024.

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

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 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 *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Audrey Huang, Adam Block, Dylan Foster, Dhruv Rohatgi, Cyril Zhang, Max Simchowitz, Jordan Ash, and Akshay Krishnamurthy. Self-improvement in language models: The sharpening mechanism. In *International Conference on Learning Representations*, volume 2025, pages 76687–76739, 2025a.

Audrey Huang, Adam Block, Qinghua Liu, Nan Jiang, Akshay Krishnamurthy, and Dylan J Foster. Is best-of-n the best of them? coverage, scaling, and optimality in inference-time alignment. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=QnjfkhrbYK.

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, 2025. URL https://arxiv.org/abs/2504.16054.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kun Lei, Huanyu Li, Dongjie Yu, Zhenyu Wei, Lingxiao Guo, Zhennan Jiang, Ziyu Wang, Shiyu Liang, and Huazhe Xu. Rl-100: Performant robotic manipulation with real-world reinforcement learning. *arXiv preprint arXiv:2510.14830*, 2025.

Qiyang Li and Sergey Levine. Q-learning with adjoint matching. *arXiv preprint arXiv:2601.14234*, 2026.

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. *arXiv preprint arXiv:2507.07969*, 2025.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

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

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *arXiv preprint arXiv:2505.05470*, 2025.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

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 *arXiv preprint arXiv:2108.03298*, 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. *arXiv preprint arXiv:2507.21053*, 2025.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

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

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, January 2024b. URL http://arxiv.org/abs/2303.05479. arXiv:2303.05479 [cs].

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International conference on machine learning*, pages 3878–3887. PMLR, 2018.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Chaoyi Pan, Giri Anantharaman, Nai-Chieh Huang, Claire Jin, Daniel Pfrommer, Chenyang Yuan, Frank Permenter, Guannan Qu, Nicholas Boffi, Guanya Shi, et al. Much ado about noising: Dispelling the myths of generative robotic control. *arXiv preprint arXiv:2512.01809*, 2025.

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

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

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. *arXiv preprint arXiv:2409.00588*, 2024.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Amrith Setlur, Matthew YR Yang, Charlie Snell, Jeremy Greer, Ian Wu, Virginia Smith, Max Simchowitz, and Aviral Kumar. e3: Learning to explore enables extrapolation of test-time compute for llms. *arXiv preprint arXiv:2506.09026*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. 2023.

Hyung Ju Suh, Max Simchowitz, Kaiqing Zhang, and Russ Tedrake. Do differentiable simulators give better policy gradients? In *International Conference on Machine Learning*, pages 20668–20696. PMLR, 2022.

Ikechukwu Uchendu, Ted Xiao, Yao Lu, Banghua Zhu, Mengyuan Yan, Joséphine Simon, Matthew Bennice, Chuyuan Fu, Cong Ma, Jiantao Jiao, et al. Jump-start reinforcement learning. In *International Conference on Machine Learning*, pages 34556–34583. PMLR, 2023.

Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. *arXiv preprint arXiv:2506.15799*, 2025.

Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning (CoRL)*, 2023.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Rosa Wolf, Yitian Shi, Sheng Liu, and Rania Rayyes. Diffusion models for robotic manipulation: A survey. *Frontiers in Robotics and AI*, 12:1606247, 2025.

Yinfei Yang, Daniel Cer, Amin Ahmad, Mandy Guo, Jax Law, Noah Constant, Gustavo Hernandez Abrego, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Multilingual universal sentence encoder for semantic retrieval, 2019. URL https://arxiv.org/abs/1907.04307.

Brent Yi, Hongsuk Choi, Himanshu Gaurav Singh, Xiaoyu Huang, Takara E Truong, Carmelo Sferrazza, Yi Ma, Rocky Duan, Pieter Abbeel, Guanya Shi, et al. Flow policy gradients for robot control. *arXiv preprint arXiv:2602.02481*, 2026.

Fan Zhang and Michael Gienger. Affordance-based robot manipulation with flow matching. *arXiv preprint arXiv:2409.01083*, 2024.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. *arXiv preprint arXiv:2505.22094*, 2025.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Zhiyuan Zhou, Andy Peng, Qiyang Li, Sergey Levine, and Aviral Kumar. Efficient online reinforcement learning fine-tuning need not retain offline data. *arXiv preprint arXiv:2412.07762*, 2024.

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*, volume 2025, pages 32343–32368, 2025.

Contents

A A Practitioner's Guide to OGPO

In this section, we enumerate key design decisions, diagnostic tools, and configurations to serve as a reference for practitioners deploying OGPO on new tasks. We defer the pseudocode to Appendix B and the low level hyperparameters to Appendix J

A.1 Key Design Decisions

While a large set of hyperparameters remain static across all our experiments, some configurations might have a large impact on OGPO's performance on tasks beyond the scope of this paper. We list each item by descending priority level denoted by its high level description followed by the variable name in the official code base.

0. Action Chunking Conventions and Critic Update The main paper denotes each action chunk $a_{t:t+h-1}$ simply as a_t for simplicity. Here we describe how this affects our computation of reward when used to train the resulting Q-function. Let us consider a standard MDP formulation where s_t is the state at current step, and $a_{t:t+h-1}$ denotes the action chunk. We follow the value backup formulation proposed in Q-chunking [Li et al., 2025], where the target uses an h-step return over the chunk and bootstraps from the value of the next action chunk at state s_{t+h}, with $a_{t+h:t+2h} \sim \pi_\theta(\cdot \mid s_{t+h})$ and $\bar{\theta}$ denoting the parameters of the target network. We use this loss to train the critic for all our off-policy methods, including OGPO, QC, DSRL, and EXPO:

$$L_{\text{critic}}(\theta) = \mathbb{E}_{s_t, a_{t:t+h}, s_{t+h} \sim \mathcal{B}} \left[\left(Q_\theta(s_t, a_{t:t+h}) - \underbrace{\sum_{t'=1}^{h} \gamma^{t'} r_{t+t'}}_{\text{effective reward}} - \gamma^h Q_{\text{targ}}(s_{t+h}, a_{t+h:t+2h}) \right)^2 \right]. \tag{A.1}$$

Algorithmic Choices

1. Behavior-cloning regularization from the success buffer (`bc_coeff`). The total objective $L_{\texttt{Total}} = L_{\texttt{PPO}} + \lambda_{\texttt{BC}} L_{\texttt{BC}}$ (Eq. (4.3)) anchors the policy to actions from $\mathcal{D}_{\text{succ}} \subseteq \mathcal{D}_{\text{roll}}$ — the subset of replay-buffer transitions belonging to successful episodes. The regularizer is asymmetric: it raises the likelihood of empirically successful actions but never lowers the likelihood of failed ones, so $L_{\texttt{BC}}$ contributes a strict lower bound on the modes $L_{\texttt{PPO}}$ is allowed to abandon. Empirically (Figure 19) this is the single most consequential modification distinguishing OGPO from OGPO+.In all experiments, we typically select $\lambda = 1.0$.

2. Conservative advantages (`adv_strategy=conservative`, Eq. (4.5)). The conservative advantage $\hat{A}_i^{\text{cons}}$ is non-zero *if and only if all M ensemble members agree on the sign of* $A_{j,m}$, in which case it takes the smallest magnitude consistent with that sign. Two consequences follow: (i) actions on which the ensemble disagrees produce no policy gradient, so the policy is updated only along directions of ensemble consensus; (ii) on directions of consensus, the magnitude is bounded by the most pessimistic Q-function, reducing the impact of outliers in the initial stages of online RL. This significantly mitigates the dip in policy evaluation and yields stable policy extraction.

3. Critic aggregation for Q_{targ} and Best-of-N (`q_agg`). As referenced in Algorithm 5, OGPO updates the critic ensemble by minimizing the Temporal Difference (TD) error. To calculate the target values,

we employ an ensemble of M target critic networks. The specific method for aggregating these target predictions is determined by the configuration flag `critic_flag`:

$$Q_{\text{targ}}(s',a') = \begin{cases} \min\{Q_{\phi_{i_1}}(s',a'), Q_{\phi_{i_2}}(s',a')\} & \texttt{critic\_flag} = \textbf{subsample} \\ \min_{i\in[M]} Q_{\phi_i}(s',a') & \texttt{critic\_flag} = \textbf{min} \\ \frac{1}{M}\sum_{i=1}^{M} Q_{\phi_i}(s',a') & \texttt{critic\_flag} = \textbf{mean} \end{cases} \tag{A.2}$$

The setting of `critic_flag` is optimized per environment (see 5). The **`min`** flag uses the minimum all Q networks, which is more aggressively curtails overestimation. The **`mean`** flag uses the mean, which is less aggressive. Many works have found **`subsample`** to be a happy medium: we take the minimum of two critic networks whose indices i_1, i_2 are sampled uniformly from the ensemble $\{1,\dots,M\}$, individually per action. Note that critic training is agnostic to the GCP structure of the policy (Mark et al. [2024]).

Eq. (A.2) aggregates the critic ensemble $\{Q_{\bar{\phi}_m}\}_{m=1}^{M}$ via $f \in \{\texttt{mean}, \texttt{min}, \texttt{subsample}\}$. Across almost all tasks, we find `subsample` being the best strategy for Q_{targ} computation when using synchronous Jax updates, but `mean` to work best using asynchronous updates. Our experiments are run on using synchronous updates. In both cases, we also find `subsample` to work optimally for selected the Best-of-N actions Eq. (4.4).

4. ODE-to-SDE conversion (`error_correct_sde_to_ode`). In **OGPO**, we add Gaussian noise of standard deviation σ_τ at each flow step to (1) ensure non-singular likelihoods thereby (2) facilitating exploration during online RL. Naively adding isotropic noise to the deterministic update $a_{t,k+1} = a_{t,k} + v_\theta(a_{t,k}, t_k \mid s_t)\Delta t$ causes distribution shift through the denoising chain, so the SDE-inferred policy visits different states than the ODE-inferred policy. Following Albergo et al. [2023], we instead use a marginal path-preserving SDE formulation that adds a score-based drift correction $\frac{\sigma_\tau^2}{2}\nabla \log p_\tau(x_\tau)$. In practice, training a separate score network (as in Liu et al. [2025]) would require modifying the BC pretraining objective, which is prohibitive for pre-trained VLAs. We instead reparameterize the score through the policy and use a tapering noise schedule $\sigma_\tau = \sigma_{\text{init}}\sqrt{1-\tau}$, which avoids the $\tau = 1$ singularity and yields the simple, numerically stable correction term

$$c = \frac{\sigma_{\text{init}}^2\left(\pi_\theta(x_\tau,\tau)\tau - x_\tau\right)}{2}. \tag{A.3}$$

See Appendix E.2 for the full derivation.

5. Warmup Phase In the code accompanying **OGPO**, we facilitate an additional *warmup*-phase to pretrain Q-functions. We provide three warmup options:

1. Warm-Start RL [Zhou et al., 2025] with Calibrated Q-Learning (CalQL) [Nakamoto et al., 2024b].
2. Q-function warmup via TD error using π_{BC} rollouts.
3. Q-functions pretrained by regressing MC returns using π_{BC} rollouts.

For the tasks considered in the paper, we generally observe warmup not being critical to policy improvement. The use of Conservative Advantages and SFT via Success Buffer have a much higher impact on **OGPO**'s training stability and sample efficiency.

Hyperparameters

1. Group size $N_{\texttt{group}}$ (`grpo_num_samples`). We rollout $N_{\texttt{group}}$ trajectories in parallel from a single s_t to compute a mean value estimate for advantage computation in Eq. (3.2). Larger $N_{\texttt{group}}$ values result in higher exploration and diversity of information points at each update at the cost of compute. We find $N_{\texttt{group}} = 32$ to be a sweet spot across all our experiments.

2. PPO clip ϵ (`clip_epsilon`). The Annealed Importance Sampling ratio ω computed in Eq. (3.2) is sensitive to small perturbations in the likelihoods of each denoising step of the GCP being used. For 10-step flow policies, we find a clipping value of $\epsilon = 0.01$ to work best for stable policy extraction. However, practitioners might need to experiment with this ratio depending on their GCP policy parameterization.

3. Update-to-data ratios We provide three key update-to-data (UTD) ratios – `utd_warmup` (number of critic updates per base policy rollout step), `utd_q`(number of critic updates per online policy rollout step), and `utd_pi`(number of actor updates per online policy rollout step). Although a UTD of 1 works across the board, they can be tweaked individually depending on the task setting.

4. Exponential Moving Average For all GCP instantiations within **OGPO**, we maintain an Exponential Moving Average (EMA) of the policy weights, denoted as θ_{EMA}. At every training step, after updating θ, we update θ_{EMA} via:

$$\theta_{\text{EMA}} \leftarrow \alpha\theta_{\text{EMA}} + (1-\alpha)\theta, \tag{A.4}$$

where α is a decay rate we typically set $\alpha = 0.995$. For **OGPO**, the EMA serves a dual purpose beyond standard stability. First, it acts as the reference policy $\pi_{\theta_{\text{old}}}$ in the PPO importance sampling ratio (Eq. (3.2)), ensuring that updates are constrained relative to a stable baseline rather than the rapidly changing online policy. Second, for the planning component in **OGPO+**, trajectories for Best-of-N ranking are sampled using $\pi_{\theta_{\text{EMA}}}$ to ensure stability in the candidate actions.

B Pseudocode

Algorithm 2 OGPO+

$\bar{\pi}_\theta, Q_{\phi_{1\ldots M}}, \mathscr{D}_{\text{roll}} \leftarrow \emptyset, \mathscr{D}_{\text{succ}} \leftarrow \emptyset$
$\theta_{\text{targ}} \leftarrow \theta$, $\phi_{\text{targ}_i} \leftarrow \phi_i \quad \forall i \in \{1, 2, \ldots M\}$
for iteration $= 1, 2, \ldots$ **do**
  Initialize state $s_{t=0} = s_0$ in M_{ENV}
  $\mathscr{T}_{\text{ep}} \leftarrow \emptyset$ `Temporary episode buffer`
  while not `done` **do**
    $(s, a, r, s', \texttt{done}) \leftarrow$ TAKE_STEP from the environment
    $\mathscr{D}_{\text{roll}} \leftarrow \mathscr{D}_{\text{roll}} \cup \{(s, a, r, s', \texttt{done})\}$
    $\mathscr{T}_{\text{ep}} \leftarrow \mathscr{T}_{\text{ep}} \cup \{(s, a, r, s', \texttt{done})\}$
    `% Update critic and policy`
    for epoch $= 1, 2, \ldots,$ `utd` **do**
      if `use_offline` **then**
        $\text{B}_{\text{itr}} \sim \{\texttt{r}_{\texttt{offline}}\mathscr{D}_{\text{off}} \cup (1 - \texttt{r}_{\texttt{offline}})\mathscr{D}_{\text{roll}}\}$
      else
        $\text{B}_{\text{itr}} \sim \mathscr{D}_{\text{roll}}$
      end if
      $\text{B}_{\text{succ}} \sim \mathscr{D}_{\text{succ}}$ **if** $\mathscr{D}_{\text{succ}} \neq \emptyset$
      UPDATEQ(B_{itr})
      UPDATEGCP($\text{B}_{\text{itr}}, \text{B}_{\text{succ}}$)
      `%Update target networks:`
      $\phi_{\text{targ},i} \leftarrow (1 - \tau)\phi_i + \tau\phi_{\text{targ},i} \forall i \in 1, \ldots, M$
      $\theta_{\text{targ}} \leftarrow (1 - \tau)\theta + \tau\theta_{\text{targ}}$
    end for
  end while
  if episode successful **then**
    $\mathscr{D}_{\text{succ}} \leftarrow \mathscr{D}_{\text{succ}} \cup \mathscr{T}_{\text{ep}}$ $\mathscr{D}_{\texttt{succ}} \subseteq \mathscr{D}_{\texttt{roll}}$
  end if
end for

return converged policy π_θ

Algorithm 3 Initialization

Function INITIALIZE($\mathscr{D}_{\text{off}}$)
{% Policy Initialization}
Pre-train GCP $\bar{\pi}_\theta^{\text{BC}}$ on $\mathscr{D}_{\text{off}}$ using BC loss $\mathscr{L}_{\text{BC}}(\theta)$
$\bar{\pi}_\theta \leftarrow \bar{\pi}_\theta^{\text{BC}}$
{% Critic Initialization}
Initialize ensemble of Q functions $Q_{\phi_{1\ldots M}}$
if `use_calql` **then**
  Pre-train $Q_{\phi_{1\ldots M}}$ on $\mathscr{D}_{\text{off}}$ using $\mathscr{L}_{\text{critic}}$ {Optional offline RL}
end if
{% Buffer Initialization}
$\mathscr{D}_{\text{roll}} \leftarrow \emptyset$
$\mathscr{D}_{\text{succ}} \leftarrow \emptyset$
{% Warmup Rollouts}
for episode $= 1, \ldots, N_{\text{warmup}}$ **do**
  Roll out $\bar{\pi}_\theta^{\text{BC}}$ in M_{ENV}, collect transitions
  $\mathscr{D}_{\text{roll}} \leftarrow \mathscr{D}_{\text{roll}} \cup \{(s, a, r, s', \texttt{done})\}_{\text{episode}}$
  if episode successful **then**
    $\mathscr{D}_{\text{succ}} \leftarrow \mathscr{D}_{\text{succ}} \cup \{(s, a, r, s', \texttt{done})\}_{\text{episode}}$
  end if
end for
if `warmup_critic` **then**
  for step $= 1, \ldots, N_{\text{critic\_warmup}}$ **do**
    $\text{B}_{\text{itr}} \sim \mathscr{D}_{\text{roll}}$
    UPDATEQ(B_{itr}) {Critic-only updates}
  end for
end if

return $\bar{\pi}_\theta, Q_{\phi_{1\ldots M}}, \mathscr{D}_{\text{roll}}, \mathscr{D}_{\text{succ}}$

Algorithm 4 Take A Step In The Environment

Function TAKE_STEP(s_t)
`done` $\leftarrow$ `False`
$a_{t,K} \sim \text{N}(0, \text{I})$
for $k = K, \ldots, 0$ **do**
  $a_{t,k-1} \leftarrow \bar{\pi}_{\theta_{\text{targ}}}(a_k, k, s_t)$
end for
$r, s_{t+1} \leftarrow$ Execute $a_{t,0}$ in environment
if s_{t+1} is terminal **then**
  `done` $\leftarrow$ `True`
end if

return $(s_t, a_{t,0}, r, s_{t+1}, \texttt{done})$

Algorithm 5 Critic Update

Function UPDATEQ(B_{itr})
$(s_t, a_{t,0}, r, s_{t+1}, \texttt{done}) \leftarrow \mathrm{B}_{\text{itr}}$
With θ frozen:
  $a_{t+1,0} \leftarrow \pi_{\theta_{\text{targ}}}(\cdot \mid s_{t+1})$
  $y \leftarrow r + \gamma \cdot \mathbb{I}[\textbf{not } \texttt{done}] \cdot Q_{\text{targ}}(s_{t+1}, a_{t+1,0})$ {Ref. Eq. A.2}
Update $\phi_{1,\ldots,M}$ via gradient descent:
  $\nabla_{\phi_i} \frac{1}{|\mathrm{B}_{\text{itr}}|} \sum_{\mathrm{B}_{\text{itr}}} \left(Q_{\phi_i}(s_t, a_{t,0}) - y\right)^2$ for $i = 1, \ldots, M$

Algorithm 6 GCP Update

Function UPDATEGCP($\mathrm{B}_{\text{itr}}, \mathrm{B}_{\text{succ}}$)
On-Policy PPO Update
  $s_t \leftarrow \mathrm{B}_{\text{itr}}$
  Sample G actions: $\{\bar{\tau}^{(g)}\}_{g=1}^{G} \sim \pi_{\theta_{\text{targ}}}(\cdot \mid s_t)$
  $\hat{A}^{\mathrm{G}} = Q_{\text{targ}}(s_t, a_{t,0}^{G}) - \mu(Q_{\text{targ}}(s_t, a_{t,0}^{G}))$
  $\omega_\theta = \frac{\prod_{k=K}^{0} \bar{\pi}_\theta^{\mathrm{G}}(a_{t,k-1} \mid a_k, k, s_t)}{\prod_{k=K}^{0} \bar{\pi}_{\theta_{\text{targ}}}^{\mathrm{G}}(a_{t,k-1} \mid a_k, k, s_t)}$
  $\mathscr{L}_{\text{PPO}}(\theta) = \mathbb{E}_{\bar{\tau} \sim \pi_{\theta_{\text{targ}}}^{\mathrm{G}}} \left[\min\left(\omega_\theta \cdot \hat{A}^{\mathrm{G}}, \text{clip}(\omega_\theta, 1-\epsilon, 1+\epsilon) \cdot \hat{A}^{\mathrm{G}}\right)\right]$
BC Update from Success Buffer
  $(s_t^{\text{succ}}, a_{t,0}^{\text{succ}}) \leftarrow \mathrm{B}_{\text{succ}}$
  $\mathscr{L}_{\text{BC}}(\theta) = \text{BCLOSS}(\bar{\pi}_\theta(\cdot \mid s_t^{\text{succ}}), a_{t,0}^{\text{succ}})$ {GCP-specific}
Combined Update
  $\mathscr{L}_{\text{total}}(\theta) = \mathscr{L}_{\text{PPO}}(\theta) + \lambda_{\text{BC}} \mathscr{L}_{\text{BC}}(\theta)$
Update θ via gradient descent on $\mathscr{L}_{\text{total}}(\theta)$

C Generative Control Policies (GCPs): A Unifying Abstraction

We propose a unifying abstraction for a broad family of popular parameterizations of control policies that we call *Generative Control Policies*, or **GCP**s. GCPs represent a stochastic policy $\pi_\theta(\cdot \mid s)$ as a series of iterative computation steps, defined by a mapping $\bar{\pi}_\theta : S \times A \times \mathbb{N}$. Given a state s_t, the policy samples $a_{t,K} \sim \bar{\pi}_\theta(\cdot \mid a_{t,k} = \emptyset, k = K, s_t)$. From then, we sample $a_{t,k-1} \sim \bar{\pi}_\theta(\cdot \mid a_{t,k}, k, s_t)$. The final action proposed is an action $a_{t,0}$. We compactly denote the distribution of this action given the observation as $a_{t,0} \sim \pi_\theta(\cdot \mid s_t)$, turning the GCP into a standard policy. Our iteration conventions are *decreasing* in K, following typical convention for diffusion models. We also drop t subscripts when clear from context.

Examples of GCPs: In addition to iterative computation, the only other requirement is that the conditional likelihoods, $\log \bar{\pi}_\theta(a_{t,k-1} = a \mid s_t, a_{t,k}, k)$ are efficiently represented. A number of popular parameterizations produce actions iteratively and satisfy this mild requirement:

- **Diffusion Policies** [Chi et al., 2023] use Denoising Diffusion Probabilistic Models (DDPMs) Ho et al. [2020]. Instantiated as an GCP, these take in pairs (s, a) as training data and iteratively add Gaussian noise to the actions through a forward process $q(a_{k+1} \mid a_k)$ and learn a function $\epsilon_\theta(a_k, k, s)$ predicting the noise added to convert x_0 to x_k. To produce an action, we sample $a_{t,K} \sim \mathrm{N}(O, \mathbf{I})$, and iteratively generate denoised samples with the following reverse process:

$$a_{k-1} \sim \bar{\pi}^{\text{DDPM}}(\cdot \mid a_k, k, s) := \mathrm{N}(\mu_k(x_k, \epsilon_\theta(a_k, k, s)), \sigma_k^2 \mathrm{I}) \tag{C.1}$$

- **Flow policies** are based on flow matching models. Given training pairs (s, a), we sample noise $z \sim \mathrm{N}(0, \mathbf{I})$, and define the interpolant $a_{(\tau)} := \tau a + (1-\tau)z$ with continuous noise index $\tau \in [0, 1]$. We then learn a velocity field $v_\theta(a_{(\tau)}, \tau, s)$, these predict $\mathbb{E}[a - z \mid s, a_{(\tau)}]$. For K discretization steps, we generate samples by initializing $a_0 \sim \mathrm{N}(0, \mathbf{I})$ and discretizing an ordinary differential equation (ODE) which reverses the noising process $a_{k-1} := a_k + \frac{1}{K} v_\theta(a_k, k/K, s)$ In its stand form, $a_{k-1} \mid a_k, s$ is deterministic. Thus, to convert a flow policy into a proper GCP, for which *likelihoods* are well-defined, we must add additional noise at each step (Reinflow [Zhang et al., 2025]). For a given choice of noise levels σ_k^2, this induces the GCP:

$$a_{k-1} \sim \bar{\pi}^{\text{FLOW}}(\cdot \mid a_k, k, s) := \mathrm{N}(v_\theta(a_k, k/K, s), \sigma_k^2 \mathbf{I}) \tag{C.2}$$

- **Minimal Iterative Policies** (MIP) are two-step flow policies which yield a performance comparable to 10-step flow policies with the natural benefit of allowing much faster inference. We defer the formal definition to Appendix C.3

The GCP formalism encompasses a number of more recent policy parameterizations as well, such as

- **Shortcut Policies** [Frans et al., 2024]: Flow models with learnable step sizes that enable variable-length generation trajectories.
- **Tokenized Autoregressive Policies** (FAST [Pertsch et al., 2025]): Policies that tokenize continuous actions in Fourier space and generate them autoregressively as discrete sequences.

In the interest of brevity, we detail the above in the Appendix C.2, and Appendix C.4 respectively. Conveniently, the GCP formalism abstracts away the details of these varying instantiations, allowing us to state all algorithms cleanly. While we have presented **OGPO** in the context of *flow-matching* policies, the algorithm is agnostic to the specific generative parameterization of the GCP, and applies directly to diffusion policies as well. Both flow-matching and score-based diffusion policies define an iterative denoising chain $a_t^K \to a_t^{K-1} \to \cdots \to a_t^0$ from a base noise distribution to the action distribution; the only difference is the parameterization of the per-step transition (a learned velocity field v_θ for flow policies versus a learned score / ϵ-prediction for diffusion). **OGPO**'s key ingredients — per-step likelihood evaluation along the denoising chain (Eq. (3.2)) and the SDE-based exploration noise correction (Appendix E.2) — are derived from generic properties of the underlying SDE and therefore carry over unchanged to a diffusion-policy GCP, provided one substitutes the appropriate noise schedule and score parameterization.

We empirically verify this in Figure 20, where we instantiate **OGPO** on top of a diffusion-policy backbone and observe consistent improvement over BC pretraining, mirroring the trends we report for flow-policy backbones in the main paper. In practice, however, we predominantly default to flow-matching policies for our main experiments: flow policies admit substantially fewer denoising steps at inference time (typically 4–10 versus 50–100 for diffusion) while achieving comparable BC performance, which directly translates into faster environment rollouts and meaningfully reduced wall-clock cost for online RL. We therefore view diffusion-policy **OGPO** as a drop-in alternative whenever the underlying VLA backbone is itself a diffusion model, and flow-policy **OGPO** as the preferred default when inference compute is a bottleneck.

C.1 OGPO with Diffusion Policies

OGPO can, in principle, be combined with any GCPs. Here, as an example, we illustrate its use in diffusion policies. We study this on the SQUARE task, where we pre-train a diffusion policy on the MH dataset and then apply online improvement with **OGPO**. As shown in Figure 20, **OGPO** successfully improves the diffusion policy to achieve mastery.

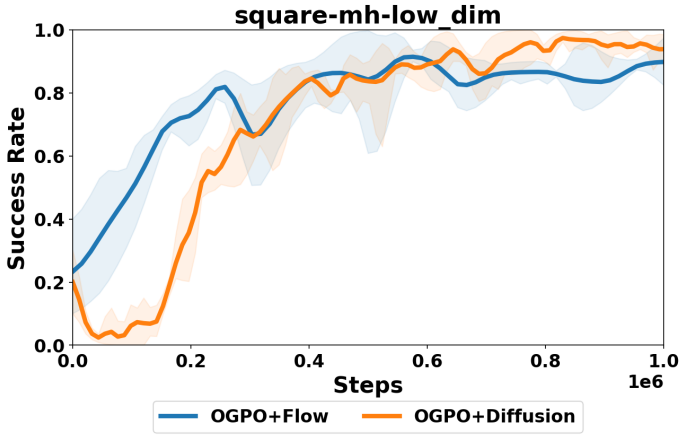


Figure 20: OGPO with diffusion policies. OGPO can successfully improve both flow policy and diffusion policy.

C.2 Shortcut Policies

Shortcut policies [Frans et al., 2024] are derived from flow-matching models conditioned on a step-size parameter d. The model $\bar{\pi}_\theta(a_t, t, d, o)$ learns to predict the next state of the flow a_{t+d} by taking a shortcut of size d. This allows the policy to function as an GCP with a variable number of refinement steps K. During pretraining, shortcut models utilize a self-consistency loss that enforces the property that one shortcut step of size $2d$ should be equivalent to two consecutive steps of size d:

$$\pi_\theta(a_t, t, 2d, o) \approx \frac{1}{2}\pi_\theta(a_t, t, d, o) + \frac{1}{2}\pi_\theta(a'_{t+d}, t+d, d, o) \tag{C.3}$$

C.3 Minimal Iterative Policy

Minimal Iterative Policies (MIP) [Pan et al., 2025] represent the simplest GCP instantiation that retains the performance benefits of flow-based policies. The key insight is that the success of generative control policies stems from combining *Stochasticity Injection* during training with *Supervised Iterative Computation*, rather than learning the distributions themselves. MIP uses only $K = 2$ denoising steps, with the first step computing $a_{t,1} \leftarrow \pi_\theta(s_t, a_{t,2} = \bar{0}, t = 0)$, then refining via $a_{t,0} \leftarrow \pi_\theta(s_t, t^\star a_{t,1}, t^\star)$. The core insight being that merely learning the conditional mean is sufficient to match the performance of complex flow-matching policies, provided the refinement steps allow the policy to adhere to the expert action manifold.

Formally, MIP optimizes the following objective during pretraining, where $t^\star = 0.9$ and $z \sim \mathcal{N}(0, I)$ is injected noise:

$$\mathcal{L}_{\text{MIP}}(\theta) = \mathbb{E}\left[\|\pi_\theta(o, I_0 = 0, t = 0) - a\|^2 + \|\pi_\theta(o, I_{t^\star}, t^\star) - a\|^2\right], \tag{C.4}$$

where $I_{t^\star}$ is the interpolant between action a and noise z.

C.4 Tokenized Autoregressive Policies

Tokenized policies, such as those using the FAST tokenizer [Pertsch et al., 2025], represent the action distribution via categorical distributions over a vocabulary of discrete tokens. FAST efficiently handles high-frequency continuous control data by applying a Discrete Cosine Transform (DCT) to action chunks, followed by quantization and Byte-Pair Encoding (BPE).

In this formulation, the GCP is an autoregressive Transformer $\bar{\pi}_\theta(z_k \mid z_{<k}, s_t)$, where z represents the sequence of discrete tokens corresponding to a compressed action chunk. The generative process iteratively samples tokens:

$$z_k \sim \text{Categorical}(\pi_\theta(\cdot \mid z_{<k}, o)) \tag{C.5}$$

Unlike diffusion or flow policies where iteration occurs in continuous action space (refining the values), here iteration occurs in the token sequence space. In particular, this slightly deviates from the GCP formulation described in the main test by requiring conditioning on the whole token sequence $z_{<k}$. However, the light likelihoods in our PPO update in Eq. (3.3) can be easily modified to handle this setting, because $p(z_{1:k}) = \prod_k p(z_k \mid z_{<k})$.

D Bi-Level MDP

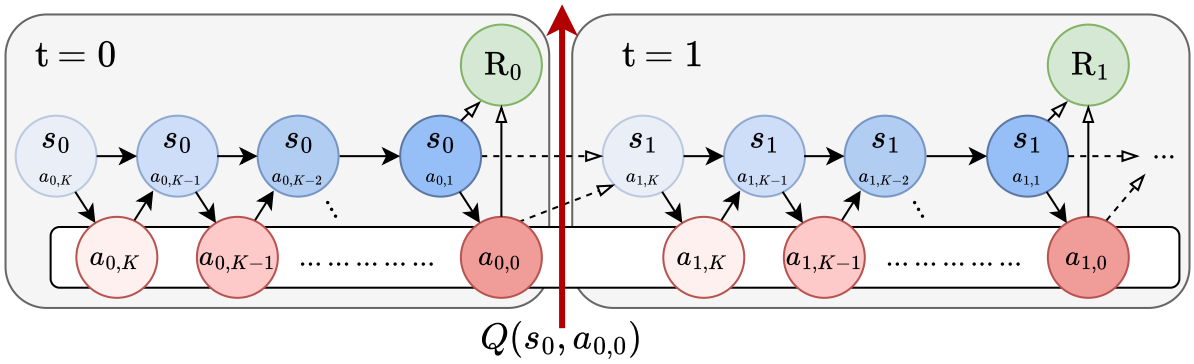


Figure 21: Bi-level (two-layer) MDP construction. Each environment step t is expanded into K inner action-generation steps indexed by $k \in \{K-1, \ldots, 0\}$. The environment transitions and rewards occur only at $k = 0$, while for $k > 0$ the state is unchanged and the inner action variable is updated.

We formulate the bi-level MDP (Figure 21), also called the two-layer MDP in [Ren et al., 2024], by embedding the action-generation dynamics into the environment dynamics. This yields an augmented MDP M_{BILEVEL} whose trajectory explicitly interleaves environment time with the K action-generation steps.

Recall the environment MDP $M_{\text{ENV}} := (S, A, P_0, P, R, \gamma)$ defined in Section 2. In M_{BILEVEL}, we index time by pairs (t, k), where t denotes the environment step and $k \in \{0, \ldots, K-1\}$ denotes the action-generation step, with $k = 0$ corresponding to executing the final action in the environment. We map (t, k) to a single time index via $\bar{t}(t, k) = tK + (K - k - 1)$, so that the sequence $\bar{t}(t, K-1), \bar{t}(t, K-2), \ldots, \bar{t}(t, 0)$ corresponds to the K generation/execution steps within environment step t. The state, action, and reward in M_{BILEVEL} are defined as

$$\bar{s}_{\bar{t}(t,k)} = (s_t, a_{t,k+1}), \quad \bar{a}_{\bar{t}(t,k)} = a_{t,k}, \quad \bar{R}_{\bar{t}(t,k)}(\bar{s}_{\bar{t}(t,k)}, \bar{a}_{\bar{t}(t,k)}) = \begin{cases} 0, & k > 0, \\ R(s_t, a_{t,0}), & k = 0. \end{cases}$$

Importantly, rewards are emitted only at indices corresponding to executing the environment action, i.e., when $a_{t,0}$ is taken. The initial distribution factorizes as $\bar{P}_0 = P_0 \otimes P_{\text{ACTION},0}$, where $s_0 \sim P_0$ is the initial environment state and $a_{0,K}$ is sampled independently from $P_{\text{ACTION},0}$, the initialization distribution for the action-generation process at $t = 0$.

Finally, the transition kernel is given by

$$\bar{P}(\bar{s}_{\bar{t}+1} \mid \bar{s}_{\bar{t}}, \bar{a}_{\bar{t}}) = \begin{cases} \delta_{(s_t, a_{t,k})}, & \bar{t} = \bar{t}(t,k),\ k > 0 \\ P(\cdot \mid s_t, a_{t,0}) \otimes P_{\text{ACTION},t+1} & \bar{t} = \bar{t}(t,k),\ k = 0 \end{cases},$$

where $P_{\text{ACTION},t}$ (for $t \geq 0$) denotes the initialization distribution for $a_{t,K}$. Intuitively, when $k > 0$, the transition advances the iterative action-generation process by moving from $(s_t, a_{t,k+1})$ to $(s_t, a_{t,k})$ while keeping the environment state fixed; when $k = 0$, it executes $a_{t,0}$ in the environment, samples $s_{t+1} \sim P(\cdot \mid s_t, a_{t,0})$, and re-initializes the next inner process by sampling $a_{t+1,K} \sim P_{\text{ACTION},t+1}$.

E Derivations

E.1 Policy Gradient Loss

An optimal policy parameterized by θ can be obtained by maximizing an objective function that computes the expected reward over a trajectory $\tau \sim \pi_\theta(\tau)$. Mathematically, $\theta^\star = \arg\max_\theta J(\theta)$, where $J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\tau)}[\omega(\tau)]$. Hence, the policy gradient objective is given as:

$$\nabla_\theta J(\theta) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau)\omega(\tau)] \tag{E.1}$$

However, there are two main challenges which make the classical PG loss formulation challenging to converge in practice. (1) Policies parameterized as neural networks can only change a little with each gradient step. (2) High variance environments require a very large number of rollouts to obtain $\pi^\star$, which is prohibitively expensive and potentially unsafe to do on real robots. As proposed by [Schulman et al., 2015], high variance can be mitigated by estimating an expectation under a distribution from an older policy $\pi_{\theta_{\text{old}}}$ using importance sampling (IS). This implies use of short horizon replay buffers where actions sampled under $\pi_{\theta_{\text{old}}}$ are reused to compute IS against π_θ. This modifies the PG objective as follows:

$$\begin{aligned} \nabla_\theta J(\theta) &= \mathop{\mathbb{E}}_{\tau \sim \pi_\theta(\tau)}\left[\frac{\pi_\theta(\tau)}{\pi_{\theta_{\text{old}}}(\tau)} \nabla_\theta \log \pi_\theta(\tau)\omega(\tau)\right] \\ &= \mathop{\mathbb{E}}_{\tau \sim \pi_\theta(\tau)}\left[\left(\sum_{t=t}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right)\left(\prod_{t=1}^{T} \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\text{old}}(a_t \mid s_t)}\right)\left(\sum_{t=t}^{T} r(s_t, a_t)\right)\right] \end{aligned} \tag{E.2}$$

However, the product of importance weights in the trajectory-level estimator leads to vanishing probability products for long horizons T. The objective is reformulated using state-action marginals. This separates the expectation over states (dependent on transition dynamics) from the expectation over actions (dependent on the policy):

$$J(\theta) = \sum_{t=1}^{T} \mathbb{E}_{s_t \sim \rho_{\theta_{\text{old}}}(s_t)}\left[\frac{\rho_\theta(s_t)}{\rho_{\theta_{\text{old}}}(s_t)} \mathbb{E}_{a_t \sim \pi_{\theta_{\text{old}}}(\cdot|s_t)}\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} r(s_t, a_t)\right]\right] \tag{E.3}$$

Calculating the state density ratio $\frac{\rho_\theta(s_t)}{\rho_{\theta_{\text{old}}}(s_t)}$ is difficult as it requires knowledge of the system dynamics. Therefore, TRPO and PPO introduce a simplification by ignoring this term. This results in a biased estimator, but the bias is negligible provided π_θ remains close to $\pi_{\theta_{\text{old}}}$. The resulting surrogate objective maximizes the probability of actions with high rewards (or advantages) relative to the old policy:

$$J(\theta) \approx \sum_{t=1}^{T} \mathbb{E}_{\substack{s_t \sim \rho_{\theta_{\text{old}}} \\ a_t \sim \pi_{\theta_{\text{old}}}}}\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} r(s_t, a_t)\right] \tag{E.4}$$

Classically, algorithms like PPO parameterize the policy $\pi_\theta(a|s)$ as a unimodal Gaussian distribution $\mathcal{N}(\mu_\theta(s), \Sigma)$. This yields a unimodal importance sampling ratio at every timestep t, which naturally struggles to model the multimodal action distributions necessary during RL exploration for complex manipulation tasks. Conversely, the total probability $\bar{\pi}_\theta(a_{t,0} \mid s_t)$ in a GCP is the product of the transition probabilities along the generation steps k. This likelihood is given as: $\pi_\theta(a_{t,0} \mid s_t) = \prod_{k=1}^{K} \pi_\theta(a_{t,k} \mid s_t)$

Substituting this into the standard PPO objective requires computing the ratio of these products. While trajectory-level importance sampling is unstable for long environment MDP chains (where $T \approx 400 - 1000$), the denoising MDP horizon of the generative process can be sufficiently short (typically $K \leq 10$)

Assuming the current policy π_θ and the reference policy (typically an Exponential Moving Average, π_{EMA}) are close, we extend the TRPO formulation to the GCP chains to compute the Annealed Importance Sampling (AIS) ratio:

$$\omega_\theta := \prod_{k=1}^{K} \frac{\pi_\theta(a^{k-1} \mid s, a^k)}{\pi_{\theta_{\text{EMA}}}(a^{k-1} \mid s, a^k)} \tag{E.5}$$

The probability of the final executed action is the joint probability of the entire chain: $\pi_\theta(a_{t,0} \mid s_t) = \prod_{k=K}^{1} \pi(a_{t,k-1} \mid a_{t,k}, s_t)$. We substitute the Monte Carlo return $\omega(\tau)$ with the advantage $\hat{A}$, which yields the final **OGPO**() objective described in Eq. (3.2). When multiplied with the advantage $\hat{A}$, the resulting gradients propagate to every step k, updating each in proportion to its contribution to the final action's probability. This end-to-end formulation ensures that generating a high-value action $a_{t,0}$ requires coherent refinement at every step $a_{t,k}$ if the GCP.

E.2 ODE-to-SDE Exploration Noise Correction

In order to have nondegenerate likelihoods, we ned to convert deterministic flow inference into a stochastic process. Naively, we could add Gaussian noise (as in Zhang et al. [2025]), but the addition of isotropic noise introduces distribution shift between the original action distribution and the noise-augmented action distribution. We note that the same approach is also adopted by Liu et al. [2025].

Specifically, we follow Albergo et al. [2023], which provides a principled conversion from ODE inference (as in standard flow models) to an SDE). Consider a continuous-time ODE

$$\mathrm{d}X_\tau = v_\theta(X_\tau, \tau)\mathrm{d}\tau, \tag{E.6}$$

where $v_\theta(x_\tau, \tau)$ is the flow velocity field. Next for a time varying diffusion standard deviation σ_τ, define an stochastic differential equation (SDE)

$$\mathrm{d}X_\tau^{\text{SDE}} = \underbrace{\left[v(X_\tau^{\text{SDE}}, \tau) + \frac{\sigma_\tau^2}{2} s(X_\tau^{\text{SDE}}, \tau) \right]}_{v^{\text{SDE}}(X_\tau^{\text{SDE}}, \tau)} \mathrm{d}\tau + \sigma_\tau \, \mathrm{d}W_\tau, \tag{E.7}$$

where $s_\tau(x) = \nabla_x \log \rho_\tau(x)$ is the score function, and where ρ_τ is the marginal distribution of X_τ.

Proposition E.1 (Albergo et al. [2023]). *For every time τ, the marginal distribution of X_τ and X_τ^{SDE} are the same.*

The key insight is that the correction in the SDE drift $v_\tau^{\text{SDE}} = v_\tau + \epsilon_\tau s_\tau$ directly offsets the effect of the Brownian drift. Furthermore, by Tweedie's formulation, the score function can be computed as

$$s(\tilde{x}_\tau, \tau) = \frac{1}{\sigma}(\mathbb{E}[Z \mid X_\tau + \sigma Z = \tilde{x}_\tau]), \quad Z \sim \mathrm{N}(0, \mathbf{I}) \tag{E.8}$$

In particular, $s_\tau = \frac{1}{\sigma} z_\tau$, where

$$z_\tau \in \arg\min_{z(\cdot)} \mathbb{E}\|z_\tau(X_\tau + \sigma Z) - Z\|^2. \tag{E.9}$$

Specialization to OGPO via discretization Given the SDE with the score correction during online RL:

$$dX_\tau = \left[\bar{\pi}_\theta(x_\tau, \tau) + \underbrace{\frac{\sigma_\tau^2}{2}\nabla \log p_\tau(x_\tau)}_{c}\right] d\tau + \sigma_\tau dW_\tau, \tag{E.10}$$

and noise schedules α_τ, β_τ, the score of the Gaussian probability path $p_\tau(x|z) \sim \mathrm{N}(x_\tau, \alpha_\tau z, \beta_\tau^2 \mathbf{I}_d)$ at timestep τ is given as

$$\nabla \log p_\tau(x|z) = -\frac{1}{\beta_\tau^2} x + \frac{\alpha_\tau}{\beta_\tau^2} z. \tag{E.11}$$

Reparameterizing the policy wrt the score function gives:

$$\bar{\pi}_\theta(x_\tau, \tau) = \left(\beta_\tau^2 \frac{\dot{\alpha_\tau}}{\alpha_\tau} - \dot{\beta_\tau}\beta_\tau\right)\nabla \log p_\tau(x_\tau) + \frac{\dot{\alpha_\tau}}{\alpha_\tau} x_\tau \tag{E.12}$$

For simplicity, we set $\alpha_\tau = \tau, \beta_\tau = 1 - \tau$. This simplifies Eq. (E.12) to

$$\nabla \log p_\tau(x|z) = \frac{\bar{\pi}_\theta(\mathbf{x}_\tau, \tau)\tau - x}{1 - \tau}. \tag{E.13}$$

Hence, the score correction term c begets

$$c = \frac{\sigma_\tau^2}{2}\nabla \log p_\tau(x|z) \tag{E.14}$$

$$= \frac{\sigma_\tau^2(\bar{\pi}_\theta(\mathbf{x}_\tau, \tau)\tau - x)}{2(1-\tau)}. \tag{E.15}$$

This reparameterization trick obviates the need for computing score function of the SDE policy, however presents an unstable divide by zero operation at $\tau = 1$, i.e. the last denoising step of the policy in practice. One way to mitigate this is to consider $\alpha_\tau = \tau, \beta_\tau = \sqrt{1-\tau^2}$ as is done by Liu et al. [2025]. However, this requires modification of the BC pretraining objective which is prohibitively expensive for pre-trained VLA models.

Therefore, we instead propose a tapering noise schedule $\sigma_\tau = \sigma_{\text{init}}\sqrt{1-\tau}$. This results in the score correction term

$$c = \frac{\sigma_{\text{init}}^2(\bar{\pi}_\theta(\mathbf{x}_\tau, \tau)\tau - x)}{2}, \tag{E.16}$$

that prevents numerical instability at the final step of the SDE rollout. We find this tapered noise schedule-based SDE flow policy to be the most stable implementation for **OGPO**. We however note that the runs presented in the paper were generated with a constant noise schedule, but our open sourced codebase provides the most optimal implementation of the SDE-flow policy.

E.3 BC on $\mathscr{D}_{\text{succ}}$ as an ELBO Barrier in Forward-KL Space

In addition to the policy gradient and pessimism terms described above, **OGPO+** also incorporates a behavior cloning (BC) loss against the success buffer $\mathscr{D}_{\text{succ}}$. We show here that this BC term serves as a tractable lower bound on the forward KL divergence $D_{\text{KL}}(\mathscr{D}_{\text{succ}} \| \pi_\theta)$, thereby aligning π_θ to the modes of successful actions and preventing the policy from dropping their probability mass.

Consider a flow policy π_θ with velocity field $v_\theta(a_\tau, \tau, s)$ trained via the linear interpolant $a_\tau = (1-\tau)\epsilon + \tau a_1$ with target $a_1 - \epsilon$. For any target action distribution q, the flow-matching loss admits the bias-variance decomposition

$$\begin{aligned}\mathscr{L}_{\text{FM}}(\theta; q) &= \mathbb{E}_{\tau,\epsilon,a_1\sim q}\big[\|v_\theta(a_\tau,\tau,s) - (a_1-\epsilon)\|^2\big] \\ &= \underbrace{\mathbb{E}\Big[\|v_\theta - v_q^\star\|^2\Big]}_{\theta\text{-optimizable}} + \underbrace{\mathbb{E}\Big[\|v_q^\star - (a_1-\epsilon)\|^2\Big]}_{\theta\text{-independent constant } C(q)},\end{aligned} \tag{E.17}$$

where $v_q^\star(a_\tau, \tau, s) := \mathbb{E}[a_1 - \epsilon \mid a_\tau, \tau, s]$ is the optimal velocity field. By Albergo et al. [2023], integrating $v_q^\star$ via the probability flow ODE in Eq. (E.6) recovers q as the terminal marginal at $\tau = 1$. The first term is therefore a tractable lower bound on the marginal forward KL:

$$\mathscr{L}_{\text{FM}}(\theta; q) - C(q) = D_{\text{KL}}(q \,\|\, \pi_\theta) \geq 0. \tag{E.18}$$

This is an ELBO in the sense that an otherwise-intractable marginal KL — the marginal densities of flow policies have no closed form — is variationally bounded by a tractable squared-error regression loss.

Instantiating this with $q = \mathscr{D}_{\text{succ}}$ recovers the BC loss on the success buffer:

$$\mathscr{L}_{\text{BC}}^{\text{succ}}(\theta) - C(\mathscr{D}_{\text{succ}}) = D_{\text{KL}}(\mathscr{D}_{\text{succ}} \,\|\, \pi_\theta). \tag{E.19}$$

Crucially, the outer expectation is taken under $\mathscr{D}_{\text{succ}}$: every successful action mode is visited at training time. If $\pi_\theta(a_{t,0}^{\text{succ}} \mid s) \to 0$ for some $a_{t,0}^{\text{succ}} \sim \mathscr{D}_{\text{succ}}$, the integrand $\log(\mathscr{D}_{\text{succ}}/\pi_\theta) \to \infty$ and the velocity-MSE penalty pulls v_θ back toward $v_{\mathscr{D}_{\text{succ}}}^\star$ at that point. This mode-preserving *barrier* property characteristic of forward KL provides regularization via the BC term. Any action mode in $\mathscr{D}_{\text{succ}}$ that π_θ tries to abandon incurs an unbounded penalty. Given the policy gradient conditioning does not strongly pull the GCP distribution against the successful modes, especially in the early training stages, π_θ retains coverage over the full support of successful behaviors throughout online RL.

F Baselines

In this section, we describe all baselines we compare to in detail. Throughout, we adopt of the action-chunking conventions of Appendix A.1.

F.1 Diffusion Policy Policy Optimization (DPPO, Ren et al. [2024])

DPPO fine-tunes diffusion policies by applying PPO directly to the bi-level MDP introduced in Appendix D. In this construction, each inner denoising step induces an explicit (Gaussian) likelihood, enabling standard policy-gradient updates on the full trajectory in M_{BILEVEL}. DPPO then instantiates the PPO clipping objective on M_{BILEVEL}.

Concretely, let $\bar{\pi}_\theta(\bar{a}_{\bar{t}} \mid \bar{s}_{\bar{t}})$ denote the policy on M_{BILEVEL} (i.e., the diffusion reverse transition at each denoising step). Given trajectories collected from $\bar{\pi}_{\theta_{\text{old}}}$ and advantage estimates $\hat{A}^{\bar{\pi}_{\theta_{\text{old}}}}$, DPPO maximizes the PPO clipped surrogate

$$\mathbb{E}_{(s_{\bar{t}},a_{\bar{t}})\sim\bar{\pi}_{\theta_{\text{old}}}}\Big[\min\Big(\frac{\bar{\pi}_\theta(a_{\bar{t}} \mid s_{\bar{t}})}{\bar{\pi}_{\theta_{\text{old}}}(a_{\bar{t}} \mid s_{\bar{t}})}\hat{A}^{\bar{\pi}_{\theta_{\text{old}}}}(s_{\bar{t}}, a_{\bar{t}}),\ \text{clip}\Big(\frac{\bar{\pi}_\theta(a_{\bar{t}} \mid s_{\bar{t}})}{\bar{\pi}_{\theta_{\text{old}}}(a_{\bar{t}} \mid s_{\bar{t}})}, 1-\epsilon, 1+\epsilon\Big)\hat{A}^{\bar{\pi}_{\theta_{\text{old}}}}(s_{\bar{t}}, a_{\bar{t}})\Big)\Big].$$

DPPO further uses an advantage estimator tailored to the bi-level structure: since rewards occur only at $\bar{t}(t,0)$, it computes environment-discounted returns across t and applies an additional denoising discount across k to downweight earlier (noisier) denoising steps.

F.2 Diffusion Steering Reinforcement Learning (DSRL, Wagenmaker et al. [2025])

DSRL improves a pretrained diffusion (or flow) policy without updating its weights by learning a policy over the *input noise space* while keeping the denoising dynamics fixed. Whereas a base diffusion policy π_{dp} samples an initial latent w_t from a fixed prior (typically $\mathcal{N}(0,\mathbf{I})$) to maps it to an executed action $a_{t,0}$ via a deterministic denoising chain (e.g., DDIM), DSRL instead formulates a *latent-action MDP* in which the fixed prior is replaced by a learnable latent policy $\pi_\psi^{\mathcal{W}}(w_t \mid s_t)$. This policy selects specific noise vectors to steer the frozen denoising process toward actions with higher expected return.

Formally, let $\pi_{\text{dp}}(s_t, w_t)$ denote the action produced by running the (frozen) denoising procedure of π_{dp} initialized at w_t, i.e., $a_{t,0} = \pi_{\text{dp}}(s_t, w_t)$. Note that if the denoising sampler is stochastic, interpret π_{dp} as inducing a conditional distribution over $a_{t,0}$ given (s_t, w_t). This induces a latent-action transition kernel

$$P^{\mathcal{W}}(s_{t+1} \mid s_t, w_t) := P\big(s_{t+1} \mid s_t, \pi_{\text{dp}}(s_t, w_t)\big),$$

and DSRL optimizes the latent policy by maximizing the discounted return in this latent-action MDP:

$$\max_\psi J(\psi) := \mathbb{E}\left[\sum_{t\geq 0} \gamma^t R\big(s_t, \pi_{\text{dp}}(s_t, w_t)\big)\right], \qquad w_t \sim \pi_\psi^{\mathcal{W}}(\cdot \mid s_t).$$

In practice, $\pi_\psi^{\mathcal{W}}$ is learned with a standard off-policy actor–critic algorithm (e.g., SAC) using transitions (s_t, w_t, r_t, s_{t+1}) collected by executing $a_{t,0} = \pi_{\text{dp}}(s_t, w_t)$ in the environment.

Optimized Variant. Our DSRL+ variant applies best-of-N filtering over steering policy actions using the Q-functions and adds a BC-loss using the success buffer to the steering policy on top of the policy graident loss.

F.3 Expressive Policy Optimization (EXPO, Dong et al. [2025])

EXPO is designed to stably fine-tune *expressive* pocilices (e.g., diffusion/flow policies) with online RL by avoiding direct value maximization through the expressive policy parameters. Instead, EXPO maintains (i) a *base* expressive policy π_{base} trained with a stable imitation (suprevised) objective, and (ii) a lightweight Gaussian *edit* policy π_{edit} that performs local action refinement toward higher Q-values. At interaction time, EXPO constructs an *on-the-fly* (OTF) policy that samples candidate actions from π_{base}, refines them with π_{edit}, and executes the candidate with the highest critic value; the same OTF selection is also used inside the TD backup.

Given $a \sim \pi_{\text{base}}(\cdot \mid s)$, EXPO samples an additive edit $\delta \sim \pi_{\text{edit}}(\cdot \mid s, a)$ and forms the refined action $\tilde{a} = a + \delta$. The OTF policy selects the better of the candidates according to the critic, $a^*(s) \in \arg\max_{a' \in \{a, \tilde{a}\}} Q_\phi(s, a')$. The edit policy is updated to increase the value of refined actions (with entropy regularization).

$$\max_{\pi_{\text{edit}}} \mathbb{E}_{(s,a)\sim\mathcal{D}, \delta\sim\pi_{\text{edit}}}\big[Q_\phi(s, a+\delta) - \alpha \log \pi_{\text{edit}}(\delta \mid s, a)\big].$$

The critic is trained by TD regression using the same OTF selection for the next-state action computed as : $\min_\phi \mathbb{E}\Big[\big(r + \gamma Q_{\phi'}(s', a^*(s')) - Q_\phi(s, a_t)\big)^2\Big]$. Finally, π_{base} is updated only through imitation-style regression (not direct Q-maximization), with value improvement coming from π_{edit} and the OTF selection.

Improve Variant. EXPO+ modifies the behavior cloning term in the standard EXPO for the "success buffer" variant described in Section 4.2.

F.4 Q-Chunking (QC, Li et al. [2025])

Recall that, in our notation, we use a single action a_t to decode an entire action-chunk in a the true environment, $a_{t:t+h-1}$. The QC algorithm proposes multiple variants. One of which, when specialized to GCPs, would require backpropagation through denoising steps, which we show leads to poor performance in Figure 16. Therefore, we opt for the other variant, which amounts to simply best-of-N inference plus behavior cloning. This variant of QC consists of three simple components:

- Learn a critic $Q(s,a)$, following the action-chunking conventions in Appendix A.1. Use this to train the critic via Eq. (2.3).
- Compute the Best-of-N action, by Q_{targ}, as following Eq. (4.4).
- Finally, we use a behavior cloning loss applied to past (s,a) pairs collected by the above planning mechanism,.

Optimized Variant. Our QC+ variant only applies BC loss to successful actions.

F.4.1 Q-Chunking v/s OGPO

Q-Chunking learns Q-functions that evaluate entire action chunks as atomic units, treating $Q(s, a_{1:H})$, where $a_{1:H}$ denotes the full action sequence over a horizon H. This formulation is agnostic to how the action chunk is generated—whether via a flow policy, a diffusion model, or direct regression. Policy improvement is guided using the Q-functions to rank a batch of actions and perform supervised fine-tuning (SFT) using BC loss on the Best-of-N actions. In contrast, OGPO explicitly leverages the iterative structure of the Generative Control Policy (GCP) by computing annealed importance sampling ratios over the denoising chain Eq. (3.2). Moreover, the advantage computation evaluates the group relative Q values over the entire action chunk and the policy gradient loss propagates through *every* denoising step k. This end-to-end formulation ensures that producing a high-value action requires coherent refinement at every GCP step, rather than treating the generation process as a black box.

F.5 ReinFlow (Zhang et al. [2025], not compared)

The ReinFlow algorithm [Zhang et al., 2025] is nearly identical to DPPO, except that it uses a flow policy as a base policy instead of Diffusion. To get non-singular likelihood rations, it augments the flow model with additional noise. However, their reported numbers are less sample efficient than DPPO (the flow sampling, however, improves *computational* efficiency), so we only use DPPO as a stronger baseline.

F.6 PA-RL (Mark et al. [2024], not compared)

The PA-RL Mark et al. [2024] algorithm is similar to QC, but includes an additional gradient ascent step $a' \leftarrow \nabla_a Q(s,a)$ to further improve actions. These gradient computations present a significant computational overhead, and perform best on TPU hardware. We found this method infeasible to run given our compute budget. Furthermore, given the instability of Q-gradients in non-smooth tasks [Suh et al., 2022], we conjecture this method would struggle in the contact-rich RoboMimic tasks.

G Understanding Exploration Behavior of OGPO

This section elaborates on the exploration dynamics of OGPO discussed in Section Section 6.1. We provide visualizations that clarify how OGPO expands the action manifold of pretrained policy distributions while maintaining stable policy improvement.

Sample Efficiency vs. Execution Efficiency In the training dynamics of OGPO, we observe two colliding optimization objectives: (1) **Sample Efficiency**: Minimizing the number of environment interactions required for policy convergence, and (2) **Execution Efficiency**: Minimizing the number of timesteps the policy takes to complete a task during inference. OGPO excels at the former via off-policy stitching, but the latter introduces unique instabilities. The discount factor $\gamma < 1$ in the Bellman equation $Q(s,a) = r + \gamma Q_{\text{targ}}(s',a')$ creates a contraction map that conditions the policy to solve tasks as quickly as possible to maximize the expected return-to-go. This causes the GCP to generate actions that could potentially maximize the speed of achieving the goal, but do not necessarily abide by physical constraints like gravity, acceleration, and robot joint position and velocity limits. This explains the oscillations in the success rate during RL-finetuning induced by rapid policy convergence via Q functions.

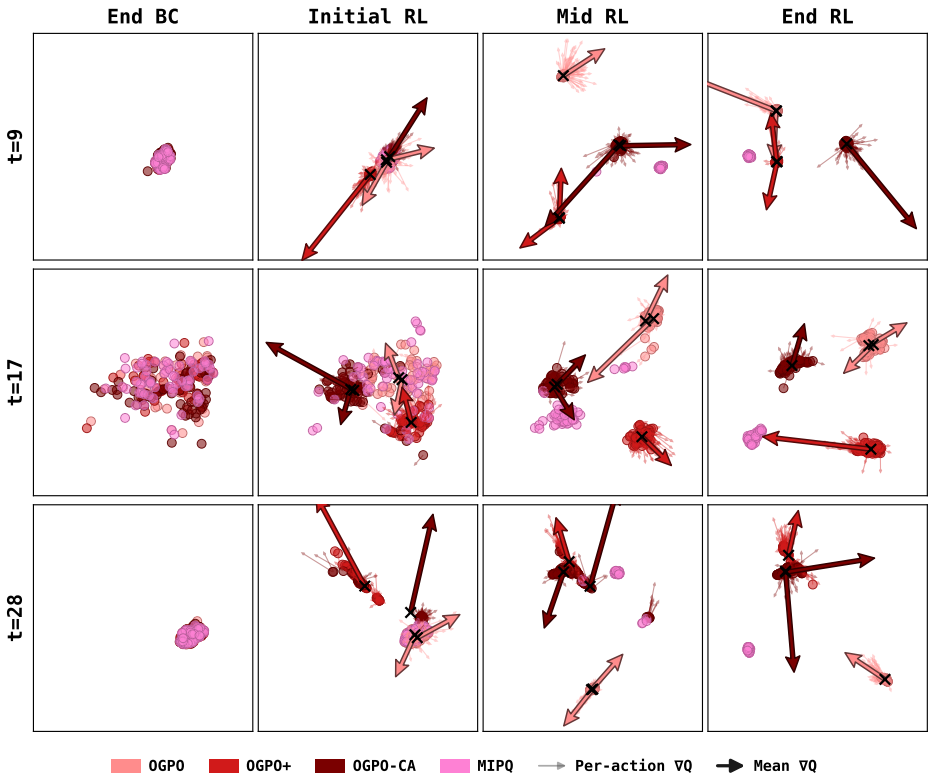


Figure 22: UMAP plot of OGPO, OGPO+, and OGPO+CA on ROBOMIMIC TOOLHANG

H Ablations and Limitations of OGPO/OGPO+

H.1 BPTT vs OGPO

The most direct way to train off-policy RL policies is to perform gradient ascent on the Q-values. Although this works for simpler policy parameterizations like Gaussian [Fujimoto et al., 2018], or Squashed Gaussian [Haarnoja et al., 2018] policies, directly using Q values to sequentially backpropagate through the GCP (also referred to as *Back Propagation Through Time (BPTT)*) can be unstable [Bengio et al., 1994]. OGPO modifies the off-policy learning paradigm for a general class of GCPs by (1) retaining the TD error loss for Q function updates, and (2) using Q functions as substitutes for Monte Carlo rollouts and computing relative advantages $\hat{A}^G$ for PPO-style updates over the entire GCP chain for the policy updates.

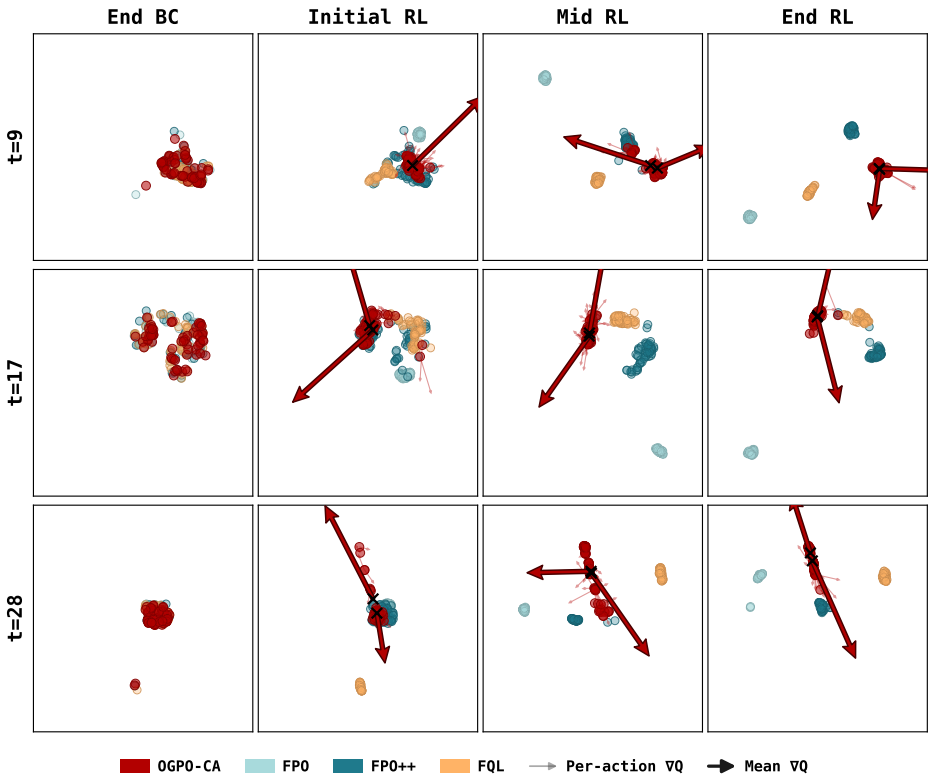


Figure 23: UMAP plot of **OGPO** comparison with various policy extraction methods on ROBOMIMIC TOOLHANG

H.2 OGPO v/s OGPO+, with and without GRPO std (σ)

GRPO formulation uses group relative advantage computation similar to **OGPO**. However, the GRPO advantage uses the standard deviation of the critic ensembles to normalize the advantage values. We found this to be empirically detrimental to **OGPO**'s success. We attribute this pattern to the sensitivity of the Annealed Importance Sampling ratio ω to very large and very small advantage values. We leave an extensive empirical validation of this sensitivity as future work.

H.3 OGPO vs Steering + Residual Ablation

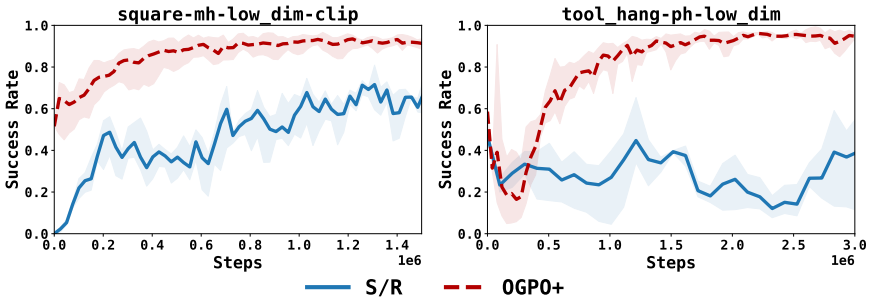


Figure 24: **OGPO+** comparison with an ablation of simultaneous steering and residual learning baseline: **S/R**

H.4 Policy Extraction Alternatives (AWR, ASPO from FPO)

OGPO separates critic learning from policy extraction: after learning Q_ϕ with the TD objective, the actor update only needs a mechanism for increasing the probability of high-advantage actions and decreasing

the probability of low-advantage actions. This makes it natural to ask whether the PPO-style extraction in OGPO is necessary, or whether simpler weighted-regression or flow-matching objectives suffice. To isolate the effect of the extraction objective, all variants below use the same replay buffer, critic update, EMA policy $\pi_{\bar{\theta}}$, and group-relative advantages $\hat{A}^{\mathrm{G}}$ as in Eq. (3.2); only the actor loss is changed.

H.4.1 Advantage-Weighted Regression and Advantage-Weighted OGPO

AWR-style extraction replaces the clipped PPO ratio with weighted flow-matching regression toward samples from the reference policy. For a sampled final action $a_0 \sim \pi_{\bar{\theta}}(\cdot \mid s)$, we define

$$w_{\mathrm{AWR}}(s, a_0) = \exp\left(\frac{\hat{A}^{\mathrm{G}}(s, a_0)}{\beta}\right), \tag{H.1}$$

where β controls how sharply the update concentrates on high-advantage samples. In our flow-policy implementation, the actor loss is not a discrete denoising-chain log-likelihood; it is a weighted conditional flow-matching objective:

$$Ł_{\mathrm{AWR}}(\theta) = \mathbb{E}_{s\sim\mathscr{B},\, a_0\sim\pi_{\bar{\theta}}(\cdot|s)}\left[\mathrm{sg}(w_{\mathrm{AWR}}(s,a_0)) \cdot \mathbb{E}_{\tau,\xi}\left\|v_\theta(x_\tau,\tau,s) - (a_0-\xi)\right\|^2\right], \tag{H.2}$$

where $\xi \sim \mathrm{N}(0, I)$, $\tau \sim \mathrm{Unif}(0,1)$, and $x_\tau = \tau a_0 + (1-\tau)\xi$. We also evaluate advantage-weighted OGPO (AW-OGPO), which preserves the same group-relative advantage computation but replaces the clipped PPO surrogate with this advantage-weighted CFM update. Empirically, these objectives are brittle on high-precision and long-horizon tasks: they can imitate high-Q samples, but do not reliably suppress bad modes when the critic is imperfect.

H.4.2 ASPO from Flow Policy Optimization

We also compare against the asymmetric trust-region objective used in Flow Policy Optimization (FPO) [Yi et al., 2026]. Instead of computing the exact denoising likelihood ratio used by OGPO, FPO constructs a surrogate ratio from the conditional-flow-matching loss:

$$\hat{r}_{\mathrm{FPO}} := \exp\left(\hat{L}_{\mathrm{CFM}}(\bar{\theta}; s, a) - \hat{L}_{\mathrm{CFM}}(\theta; s, a)\right), \tag{H.3}$$

where $\bar{\theta}$ denotes the EMA/reference policy. ASPO then applies different updates depending on the sign of the advantage. For positive advantages, it uses a PPO-style clipped objective that increases the likelihood of good actions. For negative advantages, it uses an SPO penalty with a dead zone inside the trust region:

$$\psi_{\mathrm{ASPO}}(\hat{r}_{\mathrm{FPO}}, \hat{A}^{\mathrm{G}}) = \begin{cases} \min\left(\hat{r}_{\mathrm{FPO}}\hat{A}^{\mathrm{G}},\ \mathrm{clip}(\hat{r}_{\mathrm{FPO}}, 1-\epsilon, 1+\epsilon)\hat{A}^{\mathrm{G}}\right), & \hat{A}^{\mathrm{G}} \geq 0, \\ \hat{r}_{\mathrm{FPO}}\hat{A}^{\mathrm{G}} - \dfrac{|\hat{A}^{\mathrm{G}}|}{4\epsilon}\left(\max(0,\ |\hat{r}_{\mathrm{FPO}} - 1| - \epsilon)\right)^2, & \hat{A}^{\mathrm{G}} < 0. \end{cases} \tag{H.4}$$

Thus, negative-advantage samples receive no additional SPO penalty while $\hat{r}_{\mathrm{FPO}} \in [1-\epsilon, 1+\epsilon]$; the penalty only turns on once the update moves outside the trust-region boundary, and then grows quadratically in the excess violation. Compared to OGPO, FPO avoids explicitly evaluating the full denoising-chain likelihood ratio, but this surrogate also weakens the connection between the extraction objective and the actual stochastic denoising process used during rollout.

Compared to OGPO, FPO has the appealing property that it can be implemented directly through the CFM loss, without explicitly evaluating the full denoising-chain likelihood ratio. However, this surrogate also weakens the connection between the extraction objective and the actual stochastic denoising process used during policy rollout. In our experiments, we find the FPO++'s Asymmetric trust region

(ASPO) updates to be more competitive than FPO and hence we call the off-policy version of this line of work as OFPO++. Although OFPO++ converges more generally than pure AWR, we find that it remains less stable than PPO-style extraction, especially on tasks where critic errors and low-value modes must be suppressed early in online learning.

H.5 OGPO with Flow vs. Diffusion Instantiation

While we have presented OGPO in the context of *flow-matching* policies, the algorithm is agnostic to the specific generative parameterization of the GCP and applies directly to diffusion policies as well. Both flow-matching and score-based diffusion policies define an iterative denoising chain $a_k[K] \to a_k[K-1] \to \cdots \to a_k[0]$ from a base noise distribution to the action distribution; the only difference is the parameterization of the per-step transition (a learned velocity field v_θ for flow policies versus a learned score / ϵ-prediction for diffusion). OGPO depends only on generic properties of the underlying SDE and therefore carry over unchanged to a diffusion-policy GCP, modulo the appropriate noise schedule and score parameterization.

We verify this empirically in Figure 20, where we instantiate OGPO on top of a diffusion-policy backbone and observe consistent improvement over BC pretraining, mirroring the trends reported for flow-policy backbones throughout the main paper. In practice, however, we default to flow-matching policies for our main experiments: flow policies require substantially fewer denoising steps at inference time (typically $K = 4$–10 versus $K = 50$–100 for diffusion) while achieving comparable BC performance, which directly translates to faster environment rollouts and meaningfully reduced wall-clock cost for online RL. We therefore view diffusion-policy OGPO as a drop-in alternative whenever the underlying VLA backbone is itself a diffusion model, and flow-policy OGPO as the preferred default when inference compute is a bottleneck.

I Environment Details

I.1 FRANKA-KITCHEN

The FRANKA-KITCHENbenchmark [Gupta et al., 2019] tests multi-task sequential manipulation with compositional task structure. The environment features a 9-DoF Franka robot that must manipulate 4 kitchen objects (microwave, kettle, light switch, slide cabinet) to desired goal configurations in a specific sequence. This environment is particularly challenging due to its requirement for long-horizon planning and the need to compose multiple subtasks correctly.

State and Action Spaces: The state space consists of robot joint positions, joint velocities, and object states (`state_dim` $= 60$). Actions are 9-dimensional continuous controls for the robot joints (`action_dim` $= 9$), normalized to $[-1, 1]$.

Task Horizon and Other Parameters: FRANKA-KITCHENtasks have a medium horizon of approximately 280 timesteps. We use $\gamma = 0.99$ to account for the medium-length temporal dependencies across subtasks. The action chunk size is set to $h = 4$ to provide temporal smoothness while maintaining reactivity.

Datasets: We use three offline datasets from D4RL [Fu et al., 2020]:

- KITCHEN-COMPLETE: Complete demonstrations of all 4 subtasks in the correct sequence
- KITCHEN-MIXED: Randomized subtask orders where the desired sequence is not completed sequentially
- KITCHEN-PARTIAL: Partial subtrajectories of the desired task

Reward Structure: We use a sparse reward structure with a base reward of -7. Each successful subtask completion adds +1, with the final subtask providing +3 upon success. This yields a maximum reward of 0 for completing all subtasks.

I.2 Robomimic

The ROBOMIMIC benchmark [Mandlekar et al., 2021] provides high-precision manipulation tasks that test fine-grained control and multi-step reasoning. We evaluate on three of the most challenging tasks that represent different aspects of real-world manipulation:

Square (SQUARE): A medium-horizon fine-grained insertion task requiring precise alignment and insertion of a square peg. This task tests contact-rich manipulation with tight tolerances.

- `state_dim`: 14 (robot end-effector pose, object pose)
- `action_dim`: 7 (6D end-effector control + gripper)
- Horizon: 400 timesteps
- $\gamma = 0.99$
- Action chunk size: $h = 4$
- Dataset: Multi-Human (MH) mixed proficiency

Tool Hang (TOOLHANG): A long-horizon, highly-precise multi-step insertion task requiring the robot to grasp a tool and hang it on a rack. This task demands both coarse positioning and fine-grained alignment across multiple phases.

- `state_dim`: 14
- `action_dim`: 7
- Horizon: 1000 timesteps
- $\gamma = 0.999$ (higher due to longer horizon)
- Action chunk size: $h = 8$ (larger chunks for smoother long-horizon execution)
- Dataset: Proficient-Human (PH), BC stopped at 50% success rate

Transport (TRANSPORT): A bi-manual, multi-step, long-horizon object transfer task where two robot arms must coordinate to transport an object. This tests both individual arm control and bi-manual coordination.

- `state_dim`: 28 (dual arm configuration)
- `action_dim`: 14 (7 per arm)
- Horizon: 800 timesteps
- $\gamma = 0.999$ (higher due to longer horizon)
- Action chunk size: $h = 8$
- Dataset: Multi-Human (MH) mixed proficiency

Reward Structure: All Robomimic tasks use sparse rewards: -1 for each non-successful step, with the final successful step returning 0.

Note on Hyperparameters: The different gamma values reflect the relationship between discount factor and task horizon. Longer horizon tasks (TOOLHANG, TRANSPORT) require larger gamma (0.999) to properly credit distant actions, while medium-horizon tasks (SQUARE) use smaller gamma (0.99). Similarly, longer tasks benefit from larger action chunks ($h = 8$) for smoother execution. Importantly, both gamma and chunk size are independent of action dimensionality.

I.3 Adroit Hand

The Adroit Hand benchmark tests dexterous manipulation with a 24-DoF anthropomorphic robotic hand performing high-precision, contact-rich tasks. This environment is particularly challenging due to the high-dimensional action space, under-actuated dynamics, and the need for coordinated finger movements.

We evaluate on four standard tasks:

- `AdroitHandDoor-v1`: Door opening requiring articulated finger coordination to grasp and turn a handle
- `AdroitHandHammer-v1`: Hammering a nail with precise force control and wrist articulation
- `AdroitHandPen-v1`: In-hand pen reorientation requiring complex finger gaiting
- `AdroitHandRelocate-v1`: Object relocation requiring coordinated grasping and translation

State and Action Spaces:

- `state_dim`: 45 (24 joint positions + 24 joint velocities + object state)
- `action_dim`: 24 (continuous control for each DoF)
- Actions normalized to $[-1, 1]$

Task Horizon and Temporal Parameters:

- Horizon: 200 timesteps (medium-horizon tasks)
- $\gamma = 0.95$
- Action chunk size: $h = 4$ for stabilized policy execution

Datasets: We use expert demonstration datasets provided via the D4RL/Minari interface for pre-training the base policy.

Evaluation: Following prior work, we evaluate performance using the normalized return provided by the environment, scaled to $[0, 100]$.

I.4 LIBERO

The LIBERO benchmark [Liu et al., 2023] tests vision-based, language-conditioned manipulation for multi-task learning and generalization. Unlike the previous environments, which use state-based observations, LIBERO provides pixel observations and requires following natural-language instructions, thereby testing both visual understanding and instruction-following capabilities.

Observation and Action Spaces:

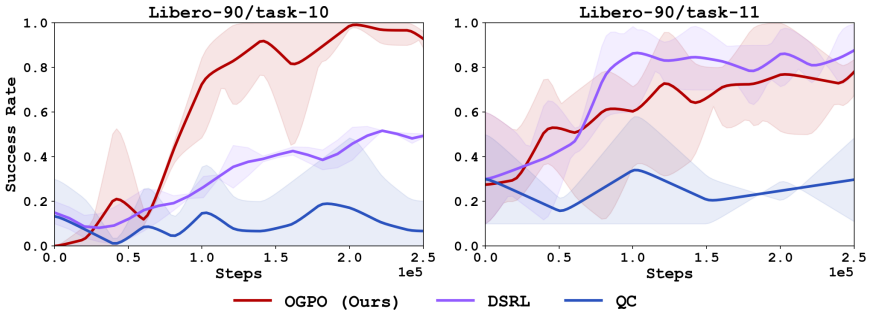


Figure 25: We compare **OGPO** with **DSRL** and **QC** on pixel-based observations and natural language guidance tasks from the LIBERO benchmark

- Observations: RGB images (128×128×3 pixels)
- `action_dim`: 7 (6D end-effector control + gripper)
- Actions normalized to $[-1, 1]$

Task Structure: LIBERO features procedurally generated tasks with natural language instructions. Tasks require understanding spatial relationships and object attributes from both visual and linguistic modalities.

Reward Structure: All Libero tasks use sparse rewards: -1 for each non-successful step, with the final successful step returning 0.

Task Horizon and Temporal Parameters:

- Horizon: 1000 timesteps (long-horizon tasks)
- $\gamma = 0.999$ (for **OGPO**, **DSRL**), 0.99 (for **QC** since we found this leads to better performance)
- Action chunk size: $h = 8$

Training and Evaluation Setup: The base policy is trained on demonstrations from 10 tasks (`task_id` $\in \{0, \ldots, 9\}$) in the Libero-90 dataset and evaluated on 2 unseen downstream tasks (`task_id` $\in 10, 11$) to test generalization capabilities. This setup explicitly tests the ability to transfer learned manipulation skills to novel task descriptions and object configurations. Since LIBERO is a language-conditioned benchmark, for both the actor and critic, we follow a widely used design from prior work [Walke et al., 2023, Nakamoto et al., 2024a]: language instructions are first processed by a frozen MUSE encoder [Yang et al., 2019] and then passed to an IMPALA encoder [Espeholt et al., 2018] with FiLM conditioning [Perez et al., 2018].

J Hyper-parameters and Initialization

J.1 Initialization and Warm Starting

OGPO accommodates two primary settings based on data availability, each with corresponding algorithmic choices for initialization. **Setting 1: Offline data available.** When an offline dataset $\mathcal{D}_{\text{off}}$ is available, we pre-train our policy π_θ^{BC} on $\mathcal{D}_{\text{off}}$ using the appropriate BC loss. The `use_offline` flag is toggled `True`, enabling offline data sampling reuse determined by the ratio $\texttt{r}_{\texttt{offline}}$.

Setting 2: No offline data (online-only). We finetune a pre-trained IGP with *no* additional demonstration data which has some small but non-trivial base success rate (>10%). The `use_offline` flag is toggled `False`.

In both settings, the online replay buffer $\mathscr{D}_{\text{roll}}$ is initialized with N_{warmup} $\bar{\pi}_\theta$ rollouts, where $\bar{\pi}_\theta \leftarrow \bar{\pi}_\theta^{\text{BC}}$. Finally, we initialize an ensemble of Q-functions $Q_{\phi_{1,\dots,M}}$ with random weights and, importantly, find that no offline RL pretraining yields the highest sample efficiency. We defer the details of the offline RL ablations to Algorithm 3.

J.2 Hyperparameters

In this section, we list all the hyper parameters we use for OGPO across different benchmarks. Table 3 shows the maximum episode lengths we use for each environment.

Environment	Max Episode Length
square	400
transport	800
tool_hang	1000
kitchen (all)	600
adroit (all)	200

Table 3: Environment maximum episode lengths

We first list the common OGPO hyper parameters. Unless otherwise stated, these remain constant throughout all our experiments. These are in Table 4.

Parameter	Default Value
lr	3e−4
actor_lr	3e−4
critic_lr	3e−4
ppo_lr	4.5e−5
tau	0.05
actor_tau	0.05
discount	0.99
batch_size	256
ppo_batch_size	256
actor_hidden_dims	(512, 512, 512, 512)
value_hidden_dims	(512, 512, 512, 512)
num_qs	10
q_agg	mean
subsample_bon	True
flow_steps	10
grpo_num_samples	32
clip_epsilon	0.01
entropy_coeff	0.0
bc_coeff	1.0
constant_noise_std	0.01
actor_scheduler	cosine
critic_scheduler	constant
actor_warmup_steps	2000
actor_decay_steps	50000
actor_end_value	2e−5
critic_warmup_steps	500
critic_decay_steps	5000
critic_end_value	0.0
actor_weight_decay	0.0
critic_weight_decay	1e−5
horizon_length	4
policy_type	flow

Table 4: OGPO agent default hyperparameters.

In Table 5, we list down all ROBOMIMIC specific hyper-parameters that are used for our experiments.

Hyperparameter	SQUARE	TOOLHANG	TRANSPORT
Training Steps			
offline_steps	500,000	500,000	1,000,000
online_steps	2,000,000	3,000,000	6,000,000
start_training	20,000	25,000	40,000
RL Hyperparameters			
horizon_length	4	8	8
discount	0.99	0.999	0.999
tau	0.05	0.05	0.05
utd_warmup	1	1	1
utd_online	1	1	1
Q-Network			
num_qs	10	10	10
q_agg	mean	mean	mean
subsample_bon	True	True	True
best_of_n	8	8	8
value_hidden_dims	(512,512,512,512)	(512,512,512,512)	(512,512,512,512,512)
BC Regularization			
use_bc_regularization	True	True	True
bc_coeff	1.0	1.0	1.0
pg_coeff	1.0	1.0	1.0
clip_bc (atmost 50% success rate)	True	True	False

Table 5: OGPO hyperparameters for Robomimic environments.

In Table 6, we list all hyper parameters we use for the various FRANKA-KITCHEN environments.

Hyperparameter	KITCHEN-COMPLETE	KITCHEN-MIXED	KITCHEN-PARTIAL
Training Steps			
offline_steps	1,000,000	1,000,000	1,000,000
online_steps	3,000,000	3,000,000	3,000,000
RL Hyperparameters			
horizon_length	4	4	4
discount	0.99	0.99	0.99
tau	0.05	0.05	0.05
utd_warmup	1	1	1
utd_online	1	1	1
Q-Network			
num_qs	10	10	10
q_agg	mean	mean	mean
subsample_bon	True	True	True
best_of_n	8	8	8
BC Regularization			
use_bc_regularization	True	True	True
bc_coeff	0.1	0.1	0.1
clip_bc	False	False	False

Table 6: OGPO hyperparameters for FRANKA-KITCHEN

In Table 7, we list all hyper parameters we use for the various `AdroitHand` environments.

Hyperparameter	Door-v1	Pen-v1	Hammer-v1	Relocate-v1
Training Steps				
offline_steps	50,000	50,000	50,000	50,000
online_steps	500,000	500,000	500,000	500,000
RL Hyperparameters				
horizon_length	4	4	4	4
discount	0.95	0.95	0.95	0.95
tau	0.05	0.05	0.05	0.05
utd_warmup	1	1	1	1
utd_online	4	4	4	4
Q-Network				
num_qs	10	10	10	10
q_agg	min	min	min	min
subsample_bon	False	False	False	False
best_of_n	8	8	8	8
BC Regularization				
use_bc_regularization	True	True	True	True
bc_coeff	1.0	1.0	1.0	1.0
clip_bc	True	True	True	True

Table 7: OGPO hyperparameters for Adroit.

In Table 8, we list all hyperparameters we use for the Libero environments.

Hyperparameter	Libero
Training	
offline_steps	50,000
online_steps	250,000
actor_tau	0.001
batch_size	64
constant_noise_std	0.01
grpo_num_samples	8
RL Hyperparameters	
horizon_length	8
discount	0.999
tau	0.05
utd_online	1
Q-Network	
num_qs	10
q_agg	mean
encoder	impala_small
value_hidden_dims	(128, 128, 128)
BC Regularization	
use_bc_regularization	False
offline_ratio	0

Table 8: OGPO hyperparameters for Libero.