OVOGO: OFFLINE VALUE-GUIDED ON-POLICY GENERATIVE OPTIMIZATION FOR FLOW-BASED VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow-based vision-language-action models can benefit from robot experience collected during deployment, but repeatedly acquiring new physical interactions for policy improvement is costly. Offline reinforcement learning provides a natural way to reuse collected experience, while flow-based policies offer an additional opportunity because fresh generation trajectories can be cheaply resampled from the current policy at replay observations. This motivates separating offline value learning from on-policy optimization of the generative process. We propose OVOGO, an offline reinforcement learning framework that realizes this separation for flow-based VLAs. A central challenge in this formulation is to learn a critic that remains grounded in logged robot experience while providing useful value signals for newly generated actions. OVOGO introduces a Return-Anchored Ranking Critic that combines in-sample distributional value learning with guarded local ranking, improving discrimination among generated actions without assigning them artificial return targets. The frozen critic then provides group-relative value signals for selected-transition optimization of fresh action candidates sampled from the current flow policy. Experiments on RoboTwin 2.0 and DexJoCo show that OVOGO improves the pretrained VLA on both benchmarks and outperforms the evaluated offline adaptation and value-guided baselines. These results suggest that offline robot experience can be repeatedly reused for value learning while inexpensive generative sampling supports policy optimization, providing an effective approach to experience-based post-training of flow-based VLAs.

1 INTRODUCTION

Vision-language-action models (VLAs) have demonstrated strong manipulation capabilities and generalization by combining large-scale, cross-task robot data with pretrained vision-language representations Kim et al. (2024); Black et al. (2024); Intelligence et al. (2025b). However, further improving these models still relies heavily on collecting high-quality demonstrations, whose acquisition is expensive and difficult to scale. In contrast, deployment and autonomous execution naturally produce large amounts of experience spanning successful, failed, and suboptimal behaviors, together with task-outcome signals that behavior cloning cannot fully exploit. Reinforcement learning (RL) provides a natural mechanism for learning from such experience, and offline RL is particularly attractive for robotics because it can repeatedly reuse previously collected interactions and their reward feedback without requiring new physical interaction for every policy update Kostrikov et al. (2021); Intelligence et al. (2025a); Wang et al. (2026b). As VLA training moves beyond imitation pretraining toward experience-based post-training, offline RL is emerging as a promising approach for improving large generative VLAs from accumulated robot experience.

Applying offline RL to flow-based VLAs, however, is not straightforward. Because actions are produced through a multi-step flow process, a central question is how value information should be translated into updates of the generative policy. Existing approaches adopt different value-to-policy interfaces. RECAP converts estimated advantages into conditioning variables and extracts an improved policy through supervised training on collected experience Intelligence et al. (2025a). This design is simple and scalable, but its improvement supervision is attached to replayed actions rather than actions freshly sampled from the current policy. Other approaches, such as LWD and Q-VGM,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

use the critic's action gradient to construct improvement targets for the flow dynamics Wang et al. (2026b;c). These methods enable more direct value-based policy improvement, but make the update depend on reliable action-gradient information from the critic around policy-generated actions. Flow policies offer a complementary opportunity because physical interactions are expensive and should be reused extensively, whereas generation trajectories are inexpensive computational samples that can be repeatedly resampled from the current policy at the same replay observation. This asymmetry suggests that physical experience and generation trajectories need not be consumed at the same rate during policy improvement.

Motivated by this observation, we decouple flow-based VLA reinforcement learning by separating offline value learning from on-policy generative policy optimization. The former learns long-horizon values from a fixed robot dataset, while the latter repeatedly samples fresh actions from the current flow policy at replay observations. The critic serves as the key interface between these two processes. Directly incorporating policy-generated actions into Bellman updates would make temporal targets depend on value estimates for actions without observed environment outcomes. Errors in these estimates, especially overestimation, can then propagate through bootstrapping and be further exploited during policy improvement. Keeping Bellman value propagation in-sample provides a more stable foundation for learning long-horizon returns because temporal targets remain grounded in replayed experience. However, restricting all critic supervision to replayed actions would leave the critic unable to reliably distinguish fresh candidates sampled from the evolving policy. The key challenge is therefore to preserve in-sample temporal value propagation while extending the critic's discriminative ability beyond replayed actions, without compromising the return-grounded value estimates on which offline learning relies.

In this paper, we propose **OVOGO** (**O**ffline **V**alue-Guided **O**n-**P**olicy **G**enerative **O**ptimization), an offline RL framework for flow-based VLAs that combines stable offline value learning with fresh on-policy optimization of the generative process. For offline value learning, OVOGO introduces a Return-Anchored Ranking Critic (RARC) that separates long-horizon return propagation from local action discrimination. RARC uses in-sample distributional value learning to propagate long-horizon values through real replay transitions without bootstrapping through policy-generated actions, and augments this return-grounded objective with local ranking supervision around replayed actions. Synthetic actions receive only relative preference supervision rather than artificial Bellman targets, while a guarded ranking objective stops enlarging Q-value gaps once sufficient separation has been achieved, preventing the ranking signal from overwhelming the return scale established by real experience. With RARC frozen, OVOGO repeatedly samples groups of fresh action candidates from the current flow policy at replay observations and constructs group-relative improvement signals directly from their scalar critic values, without differentiating through the critic. We further adopt a selected-transition flow update inspired by Flash-GRPO He et al. (2026), making on-policy generative optimization tractable for large flow-based VLAs.

In our experiments, we evaluate OVOGO on RoboTwin 2.0 Chen et al. (2025b) and DexJoCo Wang et al. (2026a). The results demonstrate that OVOGO more effectively improves flow-based VLAs from collected robot experience than existing offline adaptation and value-guided methods, while additional comparisons and ablations analyze the contributions of its critic design and generative policy optimization.

2 Related Work

2.1 Reinforcement Learning for VLA Post-Training

Recent studies have explored reinforcement learning (RL) as a post-training paradigm for improving pretrained vision-language-action models (VLAs). One line of work directly applies on-policy RL to VLAs by repeatedly collecting trajectories from the current policy and optimizing them with PPO Schulman et al. (2017), GRPO Shao et al. (2024), or related objectives Tan et al. (2025); Li et al. (2026); Chen et al. (2025a). While effective, such approaches require repeated environment rollouts, which are expensive for robotic systems; consequently, many studies rely primarily on simulated environments for RL training. Another line of work seeks to better reuse previously collected robot experience. RECAP Intelligence et al. (2025a) converts learned value information into advantage conditioning and improves the policy through supervised learning, preserving the scalability of imitation-style training but deriving its policy-improvement signal primarily from actions already

contained in the collected experience. More recent approaches such as LWD Wang et al. (2026b) and Q-VGM Wang et al. (2026c) learn explicit critics and use action gradients of the critic to guide flow-policy updates, enabling more direct value-based improvement but placing stronger requirements on the local first-order accuracy of the critic around policy-generated actions. In contrast, our method resamples candidate actions from the current flow-based VLA and uses only scalar critic values to construct relative policy-improvement signals, directly optimizing the current generative policy without differentiating through the critic.

2.2 Policy Optimization for Diffusion and Flow Policies

RL for diffusion and flow policies has been studied primarily in generic continuous-control settings. One class of methods treats the denoising or flow-generation process itself as a sequential policy and directly optimizes the generative trajectory with reinforcement learning Ren et al. (2025); Hu et al. (2025); another class avoids direct optimization of the full iterative generation process and instead transfers value information through one-step surrogate actors, value-weighted behavior learning, or regularized policy extraction Park et al. (2025); Tiofack et al. (2025); Liang et al. (2026). Most of these approaches are developed for conventional diffusion/flow control policies rather than large-scale VLAs with pretrained vision-language backbones. OGPO Patil et al. (2026) combines an off-policy critic for reusing expensive environment transitions with freshly sampled trajectories from the current generative policy, and our separation between offline value learning and on-policy generative optimization is inspired by this principle. OGPO, however, focuses on online fine-tuning of relatively lightweight generative control policies, where the replay buffer is continually refreshed through environment interaction, whereas we study offline post-training of large pretrained VLAs and explicitly address how an offline-trained critic can reliably evaluate actions generated by the evolving policy. Finally, recent work has shown that single-step policy optimization can substantially reduce the cost of reinforcement learning for large diffusion models He et al. (2026); we adopt this idea to make on-policy generative optimization practical for large flow-based VLAs.

3 Preliminaries

3.1 Flow Matching for VLA Policies

We consider a flow-based VLA policy that generates action chunks conditioned on the current VLA context Black et al. (2024). Let s_t denote the context at robot timestep t, including visual observations, the language instruction, and proprioception. We denote by $A_t = [a_t, a_{t+1}, \dots, a_{t+H-1}]$ the action chunk associated with one control transition, where H is the execution horizon. Following flow matching Lipman et al. (2022), the policy learns a continuous transport from Gaussian noise to the conditional action distribution. Given a data pair (s_t, A_t), Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$, and flow time $\tau \sim \mathcal{U}(0, 1)$, we define $x_{t,\tau} = (1-\tau)\epsilon + \tau A_t$, whose target velocity is $A_t - \epsilon$. The action expert $v_\theta(x_{t,\tau}, \tau, s_t)$ is trained with

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{(s_t, A_t),\, \epsilon,\, \tau}\left[\left\| v_\theta(x_{t,\tau}, \tau, s_t) - (A_t - \epsilon) \right\|_2^2\right]. \tag{1}$$

At inference time, sampling starts from $x_t^{[0]} \sim \mathcal{N}(0, I)$ and integrates the learned velocity field over a discretized schedule $0 = \tau_0 < \tau_1 < \cdots < \tau_K = 1$. We denote the resulting generation trajectory by $x_t^{[0]}, x_t^{[1]}, \dots, x_t^{[K]}$, where k indexes the flow-generation step and $x_t^{[K]}$ represents the generated action chunk A_t.

3.2 Offline Policy Improvement for VLAs

We formulate policy improvement at the action-chunk level. Executing A_t from context s_t produces rewards $r_t, \dots, r_{t+H-1}$ and leads to the next context s_{t+H}. The corresponding discounted chunk return is $R_t = \sum_{\ell=0}^{H-1} \gamma^\ell r_{t+\ell}$,where $\gamma \in [0, 1)$ is the per-step discount factor. The policy objective is to maximize the expected discounted return $J(\pi) = \mathbb{E}_\pi\left[\sum_{t \geq 0} \gamma^t r_t\right]$. In the offline setting, policy improvement is performed from a fixed replay dataset $\mathcal{D} = \{(s_t, A_t, R_t, s_{t+H}, d_t)\}$, where d_t indicates termination. The dataset may contain trajectories collected by one or more behavior policies. Offline reinforcement learning aims to improve the policy using only the logged experience in

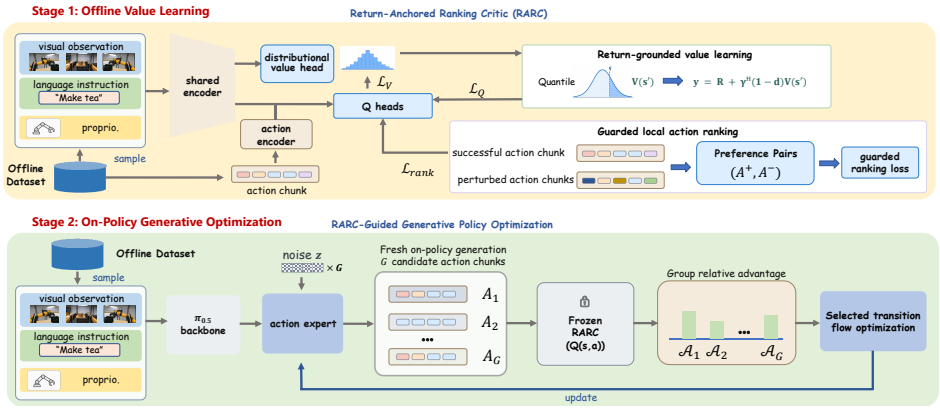

Figure 1: **Overview of OVOGO.** RARC is first trained from logged robot experience using return-grounded value learning and guarded local ranking. The frozen RARC then scores fresh policy-generated candidates to guide selected-transition flow optimization.

$\mathcal{D}$, without additional environment interaction during optimization. A practical extension is iterative offline policy improvement, where the policy is periodically deployed to collect new experience and the resulting transitions are added to the replay dataset for the next offline optimization round Intelligence et al. (2025a). Each optimization round remains offline with respect to its current replay dataset, while the accumulated dataset may expand between rounds.

4 Method

OVOGO consists of two stages. First, we train a Return-Anchored Ranking Critic (RARC) from logged robot experience using return grounded value learning and local ranking supervision, where synthetic actions provide only relative ordering constraints and never receive Bellman targets. RARC is then frozen and used to evaluate fresh action candidates sampled from the current flow policy at replay observations. Their relative critic values are converted into optimization signals for selected transition updates of the action expert, without requiring gradients through the critic.

4.1 Return-Anchored Ranking Critic

At a replay observation, the actor may generate action chunks that were never executed by the robot. Their environment outcomes are therefore unavailable, yet their critic scores determine the direction of the actor update. RARC separates two sources of supervision. Executed replay transitions establish the long-horizon return scale, while ranking perturbations provide local ordering constraints without receiving artificial return targets.

Return grounded value learning. Inspired by Distributional Implicit Value Learning (DIVL) (Wang et al., 2026b), we model the distribution of values associated with actions in replay rather than using a single scalar state value. This representation captures the heterogeneous value structure in offline robot experience and supports quantile based value extraction for temporal bootstrapping. RARC contains M ensemble members. Each member m has two scalar action value heads $Q_{m,1}$ and $Q_{m,2}$ and a categorical state value head Z_m. All members share a multimodal state encoder, while the Q heads additionally condition on the action chunk. We use Q_h, $h = 1, \ldots, 2M$, to denote individual Q heads when the member identity is irrelevant. A superscript $-$ denotes quantities evaluated using the exponential-moving-average target critic when explicitly indicated.

The categorical value head models the state conditioned distribution of Q values for actions in the replay dataset. Let $p_m(c \mid s) = \mathrm{softmax}(Z_m(s))_c$, $c = 1, \ldots, C$, denote the distribution of member m over a fixed categorical support $\mathcal{Z} = \{z_1, \ldots, z_C\}$. Within each member, we average the two target Q estimates as $\bar{Q}^-_m(s, A) = \frac{1}{2}\left(Q^-_{m,1}(s, A) + Q^-_{m,2}(s, A)\right)$. We use twin Q averaging instead of selecting the smaller estimate as in clipped double Q learning. This avoids systematically shifting the member level estimate downward before distributional value fitting. Conservatism in temporal

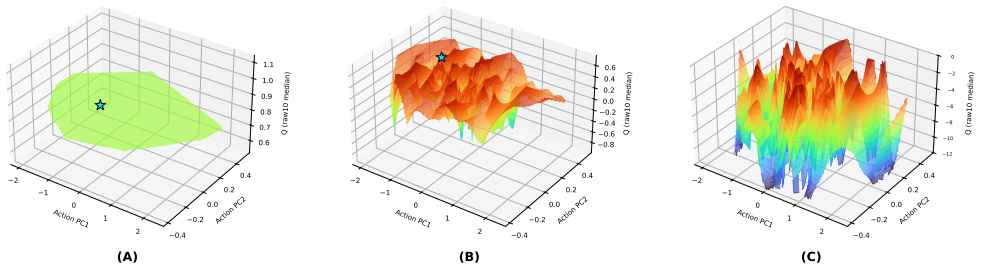

Figure 2: **Effect of local ranking on the learned action value landscape.** We visualize the median prediction across ten Q heads over locally sampled actions obtained through the same forward flow-generation process used for policy optimization, projected onto their first two principal components. The star marks the replay action. **(A)** Without ranking, the critic shows little local discrimination. **(B)** Guarded ranking introduces clear action discrimination while keeping Q value differences controlled. **(C)** Unguarded ranking produces much larger Q value variation. Raw Q values are shown with intentionally different vertical scales.

bootstrapping is handled separately through the value quantile described below. All scalar Q-head estimates are retained separately and later used for conservative evaluation of policy-generated candidates in Section 4.2.

For each (s_t, A_t) sampled from the replay dataset, we project $\bar{Q}_m^-(s_t, A_t)$ onto $\mathcal{Z}$ and use the resulting categorical distribution to supervise $p_m(\cdot \mid s_t)$

$$\mathcal{L}_V = \mathbb{E}_{\mathcal{D}}\left[\frac{1}{M}\sum_{m=1}^{M} \mathrm{CE}\left(\Pi_{\mathcal{Z}}\left[\bar{Q}_m^-(s_t, A_t)\right], p_m(\cdot \mid s_t)\right)\right]. \tag{2}$$

Following the entropy adaptive value extraction in DIVL, we adjust the bootstrap quantile according to the entropy of the learned value distribution. We compute the normalized entropy $\mathcal{H}_m(s) = -\frac{1}{\log C}\sum_{c=1}^{C} p_m(c \mid s)\log p_m(c \mid s)$ and set $\tau_m(s) = \mathrm{clip}\left(\tau_{\mathrm{base}} - \alpha\mathcal{H}_m(s), \tau_{\min}, \tau_{\max}\right)$. The corresponding state value is $V_m(s) = \mathrm{Quantile}_{\tau_m(s)}[p_m(\cdot \mid s)]$. A higher entropy leads to a lower quantile and therefore a more conservative bootstrap value, while a more concentrated distribution retains a higher quantile.

Using the chunk level transition defined in Section 3, the member specific TD target is

$$y_{m,t} = R_t + \gamma^H(1 - d_t)V_m(s_{t+H}). \tag{3}$$

Both Q heads in member m regress to the same target

$$\mathcal{L}_Q = \mathbb{E}_{\mathcal{D}}\left[\frac{1}{2M}\sum_{m=1}^{M}\sum_{j=1}^{2}\left(Q_{m,j}(s_t, A_t) - y_{m,t}\right)^2\right]. \tag{4}$$

This construction keeps temporal value propagation grounded in replay transitions with observed rewards and successor states without explicitly maximizing Q over unexecuted actions. It provides the return grounded value foundation on which we introduce local action ranking supervision next.

Guarded local action ranking. Return grounded value learning alone may provide insufficient discrimination among alternative actions at the same replay observation, while excessive ranking pressure can enlarge synthetic Q gaps beyond the scale established by replay returns, as illustrated in Figure 2.

Inspired by RankQ (Choi & Xu, 2026), we construct outcome-dependent ranking priors around replay actions. For an action chunk A_t, we sample a Gaussian direction in normalized action space and construct a mild perturbation A_t^{δ} and a stronger perturbation $A_t^{2\delta}$ along the same direction. We also construct a uniformly sampled legal action A_t^{rand} and a same task mismatched action A_t^{perm}.

For successful episodes, we use

$$A_t \succ A_t^{\delta}, \qquad A_t \succ A_t^{2\delta}, \qquad A_t \succ A_t^{\mathrm{rand}}, \qquad A_t \succ A_t^{\mathrm{perm}},$$

$$A_t^{\delta} \succ A_t^{2\delta}, \qquad A_t^{2\delta} \succ A_t^{\mathrm{rand}}.$$

Successful trajectories provide stronger evidence that their replay actions are compatible with the task. We therefore use a local smoothness prior under which smaller perturbations are more likely to preserve useful behavior. The random and mismatched actions provide coarser negative references. For failed episodes, we only impose

$$A_t \succ A_t^{\mathrm{rand}}.$$

Failed trajectories do not provide sufficient evidence for a local quality ordering among perturbations. This remaining relation encodes only a weak behavior distribution prior. The replay action comes from the behavior policy that generated the dataset, while the random action is sampled from the legal action space. It does not imply that the failed replay action has a high return.

These preferences provide relative action supervision without assigning return targets to synthetic actions. A standard pairwise softplus objective retains nonzero ranking pressure for any finite positive Q gap. Since synthetic actions have no observed environment outcomes, continued ranking pressure can push their pairwise differences away from the value scale anchored by replay returns. For Q head h and a preferred pair $A^+ \succ A^-$, we define $g_h = Q_h(s, A^+) - Q_h(s, A^-)$ and use

$$\ell_{\mathrm{rank}}(g_h) = \left[\tau_R \,\mathrm{softplus}\left(-\frac{g_h}{\tau_R}\right) - \tau_R \,\mathrm{softplus}\left(-\frac{g_{\max}}{\tau_R}\right)\right]_+ . \tag{5}$$

Here $[x]_+ = \max(x, 0)$. The temperature τ_R controls how quickly the ranking pressure decreases as a positive gap develops. The loss becomes zero once $g_h \geq g_{\max}$, so sufficiently separated action pairs receive no further ranking pressure. Our tasks use zero intermediate rewards and a terminal reward of one upon success, which places discounted returns in $[0, 1]$. We set $g_{\max} = 1$ to match the observed return scale. The guard acts only on the pairwise ranking pressure and does not clip individual Q values.

We denote the resulting ranking objective by $\mathcal{L}_{\mathrm{rank}}$. Synthetic actions participate only in this objective and never receive TD targets. The complete RARC objective is

$$\mathcal{L}_{\mathrm{RARC}} = \mathcal{L}_Q + \lambda_V \mathcal{L}_V + \lambda_R \mathcal{L}_{\mathrm{rank}}. \tag{6}$$

Return grounded value learning establishes long-horizon values from replay transitions, while guarded local action ranking provides the relative action discrimination needed for effective generative policy optimization. Once trained, RARC remains frozen during the policy optimization stage.

4.2 RARC-Guided Generative Policy Optimization

During actor optimization, observations are sampled from the fixed replay dataset. At the beginning of each actor iteration, we snapshot the current policy as the rollout policy π_{old}. For each replay observation s_b in the actor batch, the rollout policy generates a group of candidate action chunks from independent Gaussian noise,

$$\widehat{A}_{b,i} \sim \pi_{\mathrm{old}}(\cdot \mid s_b), \qquad i = 1, \ldots, G. \tag{7}$$

The frozen RARC evaluates these candidates and converts their relative values into policy optimization signals.

Group relative advantage. Policy optimization depends on the relative quality of candidates generated at the same observation. For each Q head, we first center its predictions within the candidate group,

$$\widetilde{q}_{h,b,i} = Q_h(s_b, \widehat{A}_{b,i}) - \frac{1}{G}\sum_{j=1}^{G} Q_h(s_b, \widehat{A}_{b,j}). \tag{8}$$

This removes the common value offset at the current observation while preserving the relative value differences among candidates. The categorical state value head is used only for critic learning and does not serve as the actor baseline. We then conservatively aggregate the centered scores across Q heads. A candidate receives a nonzero advantage only when all heads agree on the sign of its centered score. When all scores are positive, we use the smallest positive score. When all scores

are negative, we use the negative score closest to zero. A candidate receives zero advantage when the heads disagree on the improvement direction (Patil et al., 2026). The resulting $\mathcal{A}_{b,i}$ is detached and used as a scalar signal for policy optimization. This conservative aggregation updates the policy only along relative improvement directions consistently supported by the critic ensemble, using the most conservative magnitude among the agreeing estimates. RARC therefore interfaces with the generative policy only through scalar value feedback, and policy optimization does not require gradients through the critic.

Selected transition flow optimization. RARC evaluates a candidate after its complete action chunk has been generated. Updating the policy through an entire multistep flow trajectory, however, introduces substantial computational overhead. We therefore follow the selected transition optimization of Flash-GRPO (He et al., 2026).

For each observation s_b, we first sample a common generation transition for the entire candidate group, $k_b \sim \mathrm{Uniform}\{1, \ldots, K\}$, where transition k_b maps $x^{[k_b-1]}$ to $x^{[k_b]}$. The selected transition uses a stochastic SDE update, while the remaining generation transitions follow deterministic ODE dynamics. Each candidate is generated to completion and receives its advantage $\mathcal{A}_{b,i}$ from RARC. Policy gradients are computed only through the transition at k_b. All candidates within the same group share this selected transition, so their relative comparison is not confounded by differences across flow timesteps.

Let $\rho_{b,i}$ denote the likelihood ratio between the current policy and the rollout policy at the selected transition. Different selected timesteps induce different gradient scales through the SDE discretization. Following Flash-GRPO, we apply temporal gradient rectification with $w(k_b) = 1/\lambda(k_b)$. The clipped update is

$$\mathcal{L}_{\mathrm{policy}} = -\mathbb{E}_{b,i}\left[w(k_b)\min\left(\rho_{b,i}\mathcal{A}_{b,i}, \mathrm{clip}\left(\rho_{b,i}, 1-\varepsilon, 1+\varepsilon\right)\mathcal{A}_{b,i}\right)\right]. \tag{9}$$

Here $\lambda(k_b)$ is the timestep dependent scaling factor induced by the selected stochastic transition. The exact SDE likelihood, likelihood ratio correction, and form of $\lambda(k_b)$ follow Flash-GRPO and are provided in the Appendix. RARC evaluation of the completed action chunk provides the optimization signal, while the policy gradient is computed through a single selected transition with temporal gradient rectification, retaining the efficiency of single-transition training.

Successful behavior retention. In addition to RARC-guided policy optimization, we retain the native flow-matching objective on successful replay trajectories, denoted by $\mathcal{L}^{\mathrm{succ}}$. This auxiliary objective preserves behaviors in the replay dataset that already achieve task success.

Rather than combining the two objectives with a fixed loss coefficient, we compute their gradients separately, $g_{\mathrm{policy}} = \nabla_\theta \mathcal{L}_{\mathrm{policy}}$, $g_{\mathrm{succ}} = \nabla_\theta \mathcal{L}^{\mathrm{succ}}$, and form the actor update as

$$g_{\mathrm{actor}} = g_{\mathrm{policy}} + \lambda_{\mathrm{eff}} g_{\mathrm{succ}}. \tag{10}$$

Here λ_{eff} is determined adaptively from the global norms and cosine similarity of the two gradients, with a cap on the relative contribution of the successful-behavior gradient. RARC and the remaining VLA backbone stay frozen during policy optimization, and only the action expert is updated. The exact successful flow-matching objective, adaptive gradient-capping rule, and rollout-policy refresh procedure are provided in the Appendix.

5 Experiments

In this section, we evaluate the effectiveness of OVOGO on flow-based VLA post-training. In particular, we aim to answer the following research questions: **(1)** Compared with existing offline adaptation and value-guided methods, how effectively can OVOGO improve a pretrained VLA from the same robot experience? **(2)** Can RARC provide useful value signals for freshly generated actions, and does generative policy optimization provide additional gains over inference-time selection or guidance? **(3)** What impact do local ranking and its guard have on policy improvement?

5.1 Experimental Setup

Benchmarks and data. We evaluate on RoboTwin 2.0 (Chen et al., 2025b) and DexJoCo (Wang et al., 2026a), using ten and eleven tasks, respectively. For each task, the offline dataset contains

Table 1: Task macro success rates (%) in the standard offline setting.

Benchmark	$\pi_{0.5}$	Success BC	RECAP	Best-of-N	QGF	**OVOGO**
RoboTwin	60.7	53.8	45.1	64.6	62.3	**67.8**
DexJoCo	27.5	25.6	22.4	29.6	29.5	**35.8**

100 policy rollout episodes and 50 demonstration episodes. Within each benchmark, the actor and critic are trained jointly across all tasks using one shared model for each. All methods use the same task split, observation format, language instructions, action normalization, execution horizon, and evaluation protocol.

Baselines. We compare OVOGO with four baselines. **Success BC** adapts the action expert by behavior cloning successful trajectories from the replay dataset. **RECAP** is the value based policy adaptation method introduced with $\pi^*_{0.6}$ (Intelligence et al., 2025a). We use the Evo-RL implementation (Contributors, 2026). **Best-of-N** samples N action chunks from the policy and executes the candidate with the highest RARC score. We set $N = 8$, matching the candidate budget used by OVOGO, and use the same critic aggregation without updating the policy. **QGF** uses critic gradients to guide the flow generation process at inference time (Zhou et al., 2026). It starts from the same VLA and uses the offline trained critic without optimizing the actor from group relative candidate scores.

Training protocol. Our primary experiments follow the standard offline setting. All methods start from the same pretrained VLA $\pi_{0.5}$ and are trained on the same fixed offline dataset without additional environment interaction. For all policy updating methods, the pretrained VLM backbone is frozen and only the action expert is optimized. We additionally evaluate RECAP and OVOGO in an iterative offline setting on RoboTwin. At round k, policy optimization is performed on a fixed accumulated dataset $\mathcal{D}^{(k)}$. The updated policy then collects 100 rollout episodes per task, which are added to the dataset for the following round. The demonstration episodes remain available throughout all three rounds. Further implementation and training details are provided in the Appendix.

5.2 Main Results

Standard offline setting. Table 1 reports the performance of all methods under the standard offline setting described above. OVOGO achieves the highest success rate on both benchmarks and improves over the pretrained $\pi_{0.5}$ by 7.1 percentage points on RoboTwin and 8.3 percentage points on DexJoCo. In comparison, Success BC and RECAP do not improve over the pretrained policy under the evaluated shared multi-task setting. This contrast highlights the importance of how value information from offline robot experience is translated into policy improvement. The inference time baselines further examine whether the learned critic provides useful signals for actions generated by the policy. Best-of-N improves over $\pi_{0.5}$ by 3.9 percentage points on RoboTwin and 2.1 percentage points on DexJoCo while leaving the policy unchanged. Since Best-of-N only selects among freshly generated candidates using RARC scores, its improvement indicates that RARC captures useful relative value differences among policy-generated actions. QGF also improves over the pretrained policy by 1.6 percentage points on RoboTwin and 2.0 percentage points on DexJoCo, showing that the learned critic can also guide the flow generation process through critic gradients. OVOGO further improves over Best-of-N by 3.2 percentage points on RoboTwin and 6.2 percentage points on DexJoCo. It also exceeds QGF by 5.5 and 6.3 percentage points, respectively. Best-of-N and QGF use critic information during action generation while leaving the underlying policy unchanged. OVOGO instead uses critic feedback to update the generative policy, allowing future candidates to be sampled from an improved action distribution. The additional gains show the benefit of translating critic feedback into policy updates beyond using the critic only for inference-time selection or guidance.

Iterative offline setting. We further evaluate RECAP and OVOGO under the iterative offline setting on RoboTwin. As shown in Figure 3, OVOGO improves consistently as newly collected experience is incorporated into the offline dataset across rounds, whereas RECAP declines under the

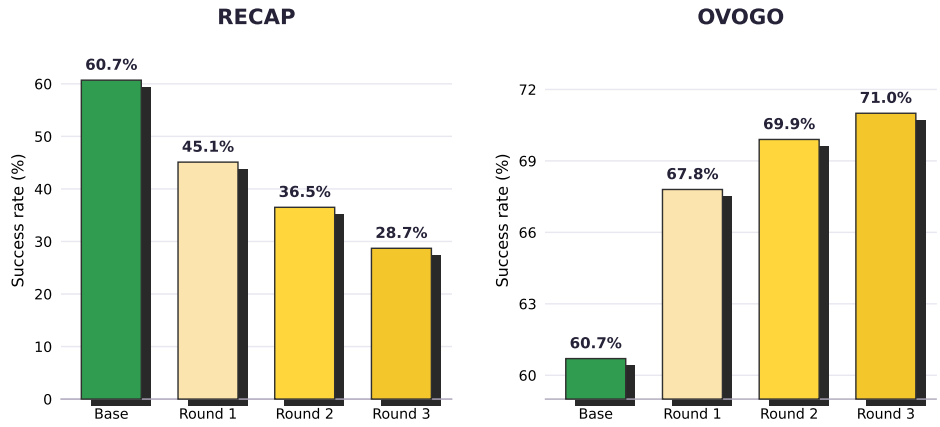

Figure 3: RoboTwin task-macro success rates for RECAP and OVOGO across three rounds of offline training and rollout collection.

Table 2: Ablation of the local ranking objective on RoboTwin under the standard offline setting.

Critic objective	Task macro success rate (%)
Return-grounded critic	61.5
+ Unguarded ranking	63.0
+ Guarded ranking (RARC)	**67.8**

same protocol. This result indicates that OVOGO can continue to benefit from accumulated robot experience over successive offline optimization rounds.

5.3 Ablation Study

We examine how local ranking and the guarded ranking design affect policy improvement. All variants use the same fixed offline dataset, return grounded value objective, and actor optimization procedure. The first variant removes local ranking and retains only the return grounded critic objective. Unguarded ranking adds the standard pairwise ranking objective with $\tau_R = 1$ and no gap guard. RARC uses guarded local ranking with $\tau_R = 0.1$ and $g_{\max} = 1$. As shown in Table 2, the return-grounded critic achieves 61.5% success. Adding unguarded local ranking increases the success rate to 63.0%. This improvement indicates that local ordering supervision helps the critic distinguish among generated action candidates and provides more useful relative value signals for policy optimization. Guarded ranking further raises the success rate to 67.8%, improving over unguarded ranking by 4.8 percentage points. The guarded design reduces ranking pressure as the preferred Q value gap increases and removes it once sufficient separation is reached. This preserves useful local action discrimination while preventing synthetic comparisons from continuously enlarging value differences beyond the scale established by replay returns. The resulting gain shows the importance of controlling ranking supervision when combining local action discrimination with return-grounded value learning.

6 Conclusion

In this work, we present OVOGO, an offline reinforcement learning framework for improving flow-based vision-language-action models from collected robot experience. OVOGO learns a return-grounded ranking critic and uses its value signals to optimize fresh actions generated by the current flow policy. Experiments on RoboTwin 2.0 and DexJoCo show that OVOGO consistently improves the pretrained VLA and achieves stronger performance than existing offline adaptation and value-guided methods. More broadly, OVOGO shows how expensive physical experience can be repeatedly reused together with inexpensive generative sampling, providing a scalable path toward experience-driven post-training of large robot policies.

AI USE STATEMENT

In this work, we used generative AI tools to assist with manuscript drafting and code inspection. We did not use these tools to collect experimental observations, generate evaluation outcomes, or produce the numerical results reported in the paper. All figures included in the submitted manuscript were drawn and finalized by the authors; no AI-generated figure was included. Additionally, we used generative AI tools for language polishing, translation, and LaTeX formatting. We reviewed all AI-assisted work, checked technical descriptions and equations against our implementation and cited sources, and checked reported results against experimental records. We take responsibility for the final content of this work, including text and claims produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide the implementation code together with the detailed hyperparameter configurations used for training and evaluation. All datasets and pretrained vision-language-action models used in our experiments are publicly available from their original sources. The main algorithmic details, training procedures, and evaluation settings are described in the paper and supplementary materials. These resources provide sufficient information for reproducing the experimental results presented in this work.

REFERENCES

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

Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, et al. πrl: Online rl fine-tuning for flow-based vision-language-action models. *arXiv preprint arXiv:2510.25889*, 8, 2025a.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025b.

Andrew Choi and Wei Xu. Rankq: Offline-to-online reinforcement learning via self-supervised action ranking. *arXiv preprint arXiv:2605.11151*, 2026.

Evo-RL Contributors. Evo-rl: Towards iterative policy improvement in real-world offline rl. `https://github.com/MINT-SJTU/Evo-RL`, 2026.

Xiaoxuan He, Siming Fu, Zeyue Xue, Weijie Wang, Ruizhe He, Yuming Li, Dacheng Yin, Shuai Dong, Haoyang Huang, Hongfa Wang, et al. Flash-grpo: Efficient alignment for video diffusion via one-step policy optimization. *arXiv preprint arXiv:2605.15980*, 2026.

Jifeng Hu, Sili Huang, Siyuan Guo, Zhaogeng Liu, Li Shen, Lichao Sun, Hechang Chen, Yi Chang, and Dacheng Tao. Decision flow policy optimization. *arXiv preprint arXiv:2505.20350*, 2025.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: a vla that learns from experience, 2025a. URL `https://arxiv.org/abs/2511.14759`.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025b.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Yang Zhaohui, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, et al. Simplevla-rl: Scaling vla training via reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 125846–125869, 2026.

Ruiming Liang, Yinan Zheng, Kexin Zheng, Tianyi Tan, Jianxiong Li, Liyuan Mao, Zhihao Wang, Guang Chen, Hangjun Ye, Jingjing Liu, et al. Dichotomous diffusion policy optimization. In *International Conference on Learning Representations*, volume 2026, pp. 61669–61690, 2026.

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

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. *arXiv preprint arXiv:2502.02538*, 2025.

Sarvesh Patil, Mitsuhiko Nakamoto, Manan Agarwal, Shashwat Saxena, Jesse Zhang, Giri Anantharaman, Cleah Winston, Chaoyi Pan, Douglas Chen, Nai-Chieh Huang, et al. Ogpo: Sample efficient full-finetuning of generative control policies. *arXiv preprint arXiv:2605.03065*, 2026.

Allen Ren, Justin Lidard, Lars Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *International Conference on Learning Representations*, volume 2025, pp. 77288–77329, 2025.

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

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.

Shuhan Tan, Kairan Dou, Yue Zhao, and Philipp Krähenbühl. Interactive post-training for vision-language-action models. *arXiv preprint arXiv:2505.17016*, 2025.

Franki Nguimatsia Tiofack, Théotime Le Hellard, Fabian Schramm, Nicolas Perrin-Gilbert, and Justin Carpentier. Guided flow policy: Learning from high-value actions in offline reinforcement learning. *arXiv preprint arXiv:2512.03973*, 2025.

Hanwen Wang, Weizhi Zhao, Xiangyu Wang, Siyuan Huang, He Lin, Boyuan Zheng, Rongtao Xu, Gang Wang, Yao Mu, He Wang, et al. Dexjoco: A benchmark and toolkit for task-oriented dexterous manipulation on mujoco. *arXiv preprint arXiv:2605.16257*, 2026a.

Yi Wang, Xinchen Li, Pengwei Xie, Pu Yang, Buqing Nie, Yunuo Cai, Qinglin Zhang, Chendi Qu, Jeffrey Wu, Jianheng Song, et al. Learning while deploying: Fleet-scale reinforcement learning for generalist robot policies. *arXiv preprint arXiv:2605.00416*, 2026b.

Ziqian Wang, Jiayu Sun, Xingjian Mao, Minqian Wang, and Yao Mu. Q-vgm: Q-guided value-gradient matching for flow-matching vla policies. *arXiv e-prints*, pp. arXiv–2606, 2026c.

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

A Implementation Details

This section provides additional implementation details of OVOGO. We first summarize the complete training procedure in Algorithm 1, and then describe the selected-transition generative policy optimization, successful-behavior retention, and the architecture and training configuration of the Return-Anchored Ranking Critic (RARC). The standard offline setting corresponds to a single offline round, whereas the iterative offline setting repeats the same critic–actor optimization procedure after augmenting the replay dataset between rounds.

Algorithm 1 OVOGO training procedure

Require: Initial replay dataset $\mathcal{D}^{(1)}$; pretrained VLA π_θ; number of offline rounds R
Ensure: Adapted policy π_θ

for $r = 1, \ldots, R$ **do**
  Stage 1: Train the return-anchored ranking critic
  for each critic update on the fixed dataset $\mathcal{D}^{(r)}$ **do**
    Sample logged transitions from $\mathcal{D}^{(r)}$
    Construct distributional value and Bellman Q targets from logged transitions using the target critic
    Construct local ranking pairs around logged actions
    Update RARC using $\mathcal{L}_{\mathrm{RARC}} = \mathcal{L}_Q + \lambda_V \mathcal{L}_V + \lambda_R \mathcal{L}_{\mathrm{rank}}$
    Update the target critic by exponential moving averaging
  end for
  Freeze RARC
  Stage 2: Optimize the action expert
  Set $\pi_{\mathrm{old}} \leftarrow \pi_\theta$
  for each actor update on the fixed dataset $\mathcal{D}^{(r)}$ **do**
    Sample a batch of replay observations
    for each replay observation s_b in the batch **do**
      Sample a common $k_b \sim \mathrm{Uniform}\{1, \ldots, K\}$ for the candidate group
      Generate G complete chunks under π_{old}, with only transition k_b stochastic
      Score the completed chunks with the frozen RARC scalar Q heads
      Construct detached group-relative advantages using conservative ensemble aggregation
    end for
    Compute the selected-transition Flash-GRPO gradient g_{policy}
    Compute the success-only flow-matching gradient g_{succ}
    Compute λ_{eff} and combine $g_{\mathrm{actor}} = g_{\mathrm{policy}} + \lambda_{\mathrm{eff}} g_{\mathrm{succ}}$
    Clip g_{actor} and update only the action expert
    Refresh $\pi_{\mathrm{old}} \leftarrow \pi_\theta$
  end for
  if $r < R$ **then**
    Collect rollout episodes $\mathcal{E}^{(r+1)}$ with the adapted policy
    Set $\mathcal{D}^{(r+1)} \leftarrow \mathcal{D}^{(r)} \cup \mathcal{E}^{(r+1)}$
  end if
end for
return π_θ

A.1 Selected-Transition Generative Policy Optimization

We adopt the selected-transition optimization strategy of Flash-GRPO (He et al., 2026) to efficiently optimize the multi-step flow-generation process. The central idea is to evaluate a completed generated action chunk while propagating policy gradients through only one stochastic transition of its generation trajectory.

For each replay observation s_b, the rollout policy π_{old} generates a group of G candidate action chunks from independent Gaussian noise,

$$x_{b,i}^{[0]} \sim \mathcal{N}(0, I), \qquad \widehat{A}_{b,i} = x_{b,i}^{[K]}, \qquad i = 1, \ldots, G, \tag{11}$$

where K denotes the number of flow transitions.

Following the iso-temporal grouping principle of Flash-GRPO, all candidates associated with the same replay observation share a common selected generation step,

$$k_b \sim \mathrm{Uniform}\{1, \ldots, K\}. \tag{12}$$

The selected transition maps $x_{b,i}^{[k_b-1]}$ to $x_{b,i}^{[k_b]}$. Sharing the same selected timestep within a candidate group avoids confounding relative candidate comparisons with differences in generation time.

Stochastic selected transition. To follow the notation of Flash-GRPO, we use t for the reverse-time parameterization in this subsection. This variable is distinct from the forward flow-time variable τ used in the flow-matching preliminaries. Following the SDE formulation used in Flash-GRPO, a deterministic flow ODE can be transformed into a stochastic transition while preserving the same

marginal generation process. At flow time t, the selected transition is written as

$$\boldsymbol{x}_{t-\Delta t} = \boldsymbol{x}_t + \left[\boldsymbol{v}_\theta(\boldsymbol{x}_t, t) + \frac{\sigma_t^2}{2t}\Big(\boldsymbol{x}_t + (1-t)\boldsymbol{v}_\theta(\boldsymbol{x}_t, t)\Big)\right]\Delta t + \sigma_t\sqrt{\Delta t}\,\boldsymbol{\epsilon}, \tag{13}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$, $\boldsymbol{v}_\theta$ is the velocity field predicted by the action expert, and σ_t controls the stochasticity of the transition.

The conditional mean of the selected transition is therefore

$$\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) = \boldsymbol{x}_t + \left[\boldsymbol{v}_\theta(\boldsymbol{x}_t, t) + \frac{\sigma_t^2}{2t}\Big(\boldsymbol{x}_t + (1-t)\boldsymbol{v}_\theta(\boldsymbol{x}_t, t)\Big)\right]\Delta t. \tag{14}$$

Accordingly, the selected-transition likelihood is

$$p_\theta\left(\boldsymbol{x}_{t-\Delta t} \mid \boldsymbol{x}_t, s_b\right) = \mathcal{N}\left(\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \sigma_t^2 \Delta t\, I\right). \tag{15}$$

All remaining generation steps follow the corresponding deterministic flow dynamics. Importantly, each candidate is still generated to completion before being evaluated by the frozen RARC. The critic thus assigns value to the complete generated action chunk, while the policy gradient is computed only through the selected transition.

Selected-transition likelihood ratio. Let $\ell_{\theta,b,i}$ and $\ell_{\text{old},b,i}$ denote the log likelihoods assigned to the selected transition under the current and rollout policies, respectively. Let D_f denote the event dimension of the selected flow transition. We first form the dimension-normalized log-likelihood ratio

$$\Delta_{b,i} = \frac{\ell_{\theta,b,i} - \ell_{\text{old},b,i}}{D_f}. \tag{16}$$

For numerical stability, the normalized log ratio is clipped before exponentiation,

$$\rho_{b,i} = \exp\left[\text{clip}\left(\Delta_{b,i}, -c_\rho, c_\rho\right)\right], \tag{17}$$

where c_ρ is the log-ratio clipping threshold listed in Table 4. We use dimension normalization and log-ratio clipping as numerical stabilization for the high-dimensional selected transition.

Temporal gradient rectification. A stochastic SDE transition introduces a timestep-dependent scaling factor into the policy gradient. Following Flash-GRPO, this factor is

$$\lambda(t) = \frac{\sqrt{\Delta t}}{\sigma_t} + \frac{\sigma_t\sqrt{\Delta t}(1-t)}{2t}. \tag{18}$$

Without correction, different selected flow timesteps can therefore produce substantially different gradient magnitudes even when their advantages have comparable scales. We apply temporal gradient rectification using

$$w(t) = \frac{1}{\lambda(t)}. \tag{19}$$

Let $\mathcal{A}_{b,i}$ denote the detached group-relative advantage provided by RARC. The selected-transition policy objective is

$$\mathcal{L}_{\text{policy}} = -\mathbb{E}_{b,i}\Bigg[w(t_{k_b})\min\Big(\rho_{b,i}\mathcal{A}_{b,i},\ \text{clip}(\rho_{b,i}, 1-\varepsilon, 1+\varepsilon)\mathcal{A}_{b,i}\Big)\Bigg]. \tag{20}$$

Only the selected transition contributes policy-likelihood gradients. The remaining generation trajectory is used to construct the complete candidate action but is not differentiated through. Similarly, RARC is frozen during actor optimization, and the critic-derived advantage is detached before the policy update.

The rollout policy π_{old} is refreshed from the current action expert after each policy update. This maintains a locally consistent reference distribution for the PPO-style likelihood ratio.

A.2 Successful-Behavior Retention and Gradient Control

RARC-guided policy optimization encourages the action expert to shift probability toward candidates preferred by the critic. At the same time, we retain behaviors in the offline replay dataset that already achieve task success by applying the native flow-matching objective to successful trajectories.

Let $\mathcal{D}_{\text{succ}}$ denote successful replay samples. For a successful logged action chunk A_t, we sample $\epsilon \sim \mathcal{N}(0, I)$ and flow time $\tau \sim \mathcal{U}(0, 1)$ and construct

$$x_\tau = (1 - \tau)\epsilon + \tau A_t. \tag{21}$$

The corresponding successful-behavior flow-matching objective is

$$\mathcal{L}_{\text{FM}}^{\text{succ}} = \mathbb{E}_{(s_t, A_t) \sim \mathcal{D}_{\text{succ}},\, \epsilon, \tau} \left[\| v_\theta(x_\tau, \tau, s_t) - (A_t - \epsilon) \|_2^2 \right]. \tag{22}$$

Rather than combining the two objectives through a fixed scalar loss weight, we compute their gradients separately. Let

$$g_{\text{policy}} = \nabla_\theta \mathcal{L}_{\text{policy}}, \qquad g_{\text{succ}} = \nabla_\theta \mathcal{L}_{\text{FM}}^{\text{succ}}. \tag{23}$$

A fixed auxiliary coefficient can allow the flow-matching gradient to dominate the critic-guided policy update when the two objectives have substantially different gradient scales. We therefore use a nominal coefficient λ_0 together with an adaptive gradient cap.

Define the global gradient norms

$$a = \| g_{\text{policy}} \|_2, \qquad b = \| g_{\text{succ}} \|_2, \tag{24}$$

and, when both gradients have nonzero norm, their cosine similarity

$$c = \frac{\langle g_{\text{policy}}, g_{\text{succ}} \rangle}{ab}. \tag{25}$$

For numerical stability, c is clipped to the interval $[-1, 1]$.

Let $\rho_{\text{BC}} \in (0, 1)$ denote the target maximum contribution of the weighted successful-behavior gradient relative to the norm of the combined actor gradient. We compute

$$\xi = \frac{\rho_{\text{BC}}^2 c + \rho_{\text{BC}} \sqrt{\max\left(0,\, 1 - \rho_{\text{BC}}^2 + \rho_{\text{BC}}^2 c^2\right)}}{1 - \rho_{\text{BC}}^2}, \tag{26}$$

followed by

$$\lambda_{\text{cap}} = \frac{\xi a}{b + \epsilon_g}, \tag{27}$$

and

$$\lambda_{\text{eff}} = \min\left(\lambda_0, \max(0, \lambda_{\text{cap}})\right). \tag{28}$$

The final actor gradient is then

$$g_{\text{actor}} = g_{\text{policy}} + \lambda_{\text{eff}} g_{\text{succ}}. \tag{29}$$

For non-degenerate gradients, this construction enforces the target cap

$$\frac{\| \lambda_{\text{eff}} g_{\text{succ}} \|_2}{\| g_{\text{policy}} + \lambda_{\text{eff}} g_{\text{succ}} \|_2} \leq \rho_{\text{BC}}. \tag{30}$$

Thus, ρ_{BC} is an upper-bound target rather than a fixed mixture coefficient. When the successful-behavior gradient is small, the nominal coefficient λ_0 is retained. When it becomes large, its effective weight is reduced so that the auxiliary update does not dominate the RARC-guided policy gradient.

When $b \leq \epsilon_g$, we retain the nominal coefficient λ_0. If $a = 0$ while $b > \epsilon_g$, the adaptive rule yields $\lambda_{\text{eff}} = 0$. These cases avoid unstable scaling when one of the two gradient components is numerically negligible.

Only successful replay trajectories contribute to this auxiliary flow-matching objective. Failed trajectories do not receive behavior cloning supervision, and we do not introduce an additional action-space mean-squared-error cloning objective. The two gradient components are explicitly combined before gradient clipping and a single optimizer step.

A.3 RARC Critic Architecture and Training

RARC uses frozen pretrained visual and language encoders together with a lightweight trainable critic. The language backbone is Gemma 3 270M, and the vision backbone is SigLIP2-So400M with 224×224 visual inputs. The pretrained Gemma 3 and SigLIP2 parameters remain frozen during critic optimization; only the critic-specific projection layers and prediction heads are trained. The trainable critic core and encoder projection contain approximately 7.78M parameters.

For RoboTwin, the critic receives three visual observations corresponding to the base camera and the two wrist cameras, together with the language instruction and robot proprioceptive state. The same critic design is used for DexJoCo with the benchmark-specific observation interface.

RARC contains $M = 5$ ensemble members. Each member contains two scalar action-value heads and one categorical state-value head, yielding ten raw scalar Q estimates in total. The categorical value head uses $C = 201$ atoms on the fixed support $[-0.1, 1.1]$. We retain the ten Q estimates separately when evaluating policy-generated candidates.

EMA target critic. For training stability, we maintain an exponential-moving-average copy of the complete critic, including the trainable encoder projection, scalar Q heads, and categorical value heads. The target Q predictions are used to construct the distributional value supervision, while the categorical value distribution used for TD bootstrapping is evaluated with the EMA target critic. The target critic is updated after every critic optimization step with coefficient $\eta_{\mathrm{tgt}} = 0.005$.

The return-grounded distributional value objective, entropy-adaptive quantile extraction, scalar Q objective, and guarded local ranking objective are defined in Section 4.1; we do not repeat those formulations here. Their implementation-specific numerical settings are summarized in Table 4.

B Experimental Details

B.1 Environments and Policy Initialization

RoboTwin 2.0. We evaluate on ten tasks from RoboTwin 2.0 (Chen et al., 2025b). Each task uses visual observations, a language instruction, and robot proprioception. The policy predicts action chunks of length $H = 50$.

We initialize the policy from the publicly released $\pi_{0.5}$ clean50 checkpoint.[1]

The RoboTwin environment action representation has dimension 50×14, while the $\pi_{0.5}$ action expert internally operates on a 50×32 flow representation before conversion to environment actions.

DexJoCo. We evaluate on eleven tasks from DexJoCo (Wang et al., 2026a), including five bimanual tasks and six single-arm tasks. The multi-task policy predicts action chunks of length $H = 30$. Single-arm and bimanual tasks share a unified multi-task policy interface using the benchmark-specific state and action representations.

We initialize from the official multi-task $\pi_{0.5}$ checkpoint.[2] The released checkpoint is provided in JAX/Orbax format. We convert the checkpoint to PyTorch before training, including the vision-language and action-expert parameters and the corresponding normalization statistics. The converted model is used consistently for all DexJoCo comparisons.

Except for benchmark-specific observation and action interfaces, we use the same OVOGO training configuration for RoboTwin and DexJoCo.

B.2 Task Suites

Table 3 lists all tasks used in our experiments together with a short description of the required manipulation. The descriptions summarize the task semantics indicated by the official benchmark task definitions and language instructions.

[1] https://huggingface.co/madokalif/pi05-robotwin2-clean50-sft

[2] https://huggingface.co/DexJoCo/DexJoCo-Pi05/tree/main/pi05_dexjoco_multi_task

Table 3: Task suites used for RoboTwin 2.0 and DexJoCo.

Benchmark	Task	Type	Task description
RoboTwin 2.0	adjust_bottle	Bimanual	Grasp the bottle with the side-appropriate arm and move it to the corresponding elevated target side.
	click_alarmclock	Bimanual	Press the top button of the alarm clock with the arm corresponding to its location.
	click_bell	Bimanual	Press the top of the bell with the arm corresponding to its location.
	dump_bin_bigbin	Bimanual	Lift the small tabletop bin and pour its contents into the large bin.
	handover_mic	Bimanual	Grasp the microphone with one arm and hand it over to the opposite arm.
	open_laptop	Bimanual	Grasp the laptop lid and open it beyond the required hinge angle.
	pick_dual_bottles	Bimanual	Grasp the two bottles and move them to their corresponding left and right elevated targets.
	place_a2b_right	Bimanual	Grasp object A, place it to the right of object B, and release it.
	place_object_stand	Bimanual	Grasp the target object, place it on the display stand, and release it.
	press_stapler	Bimanual	Press down on the stapler with the arm corresponding to its location.
DexJoCo	bimanual_assembly	Bimanual	Grasp the tray with the left hand and the peg with the right hand, then insert the peg into the hole.
	bimanual_hanoi	Bimanual	Execute the final two Tower-of-Hanoi moves: medium disk middle-to-right, then small disk left-to-right.
	bimanual_microwave_cook	Bimanual	Open the microwave, place the food inside, close the door, and press the start button.
	bimanual_photograph	Bimanual	Hold the camera with the left hand, align it with the logo, and press the shutter with the right hand.
	bimanual_unlock_ipad	Bimanual	Grasp the iPad and enter the password 123 to unlock it.
	click_mouse	Single-arm	Move the mouse to the purple mouse pad and click the left mouse button.
	fold_glasses	Single-arm	Fold the glasses and place them into the case.
	hammer_nail	Single-arm	Use the hammer to drive the nail into the wooden board.
	pick_bucket	Single-arm	Place the boxed food into the bucket and then lift the bucket.
	pinch_tongs	Single-arm	Grasp the tongs and perform three consecutive open-close motions.
	water_plant	Single-arm	Grasp the watering can and apply water to the plant.

B.3 Training Hyperparameters

Table 4 summarizes the principal OVOGO hyperparameters. Unless otherwise stated, the same optimization configuration is used for both benchmarks.

Table 4: Main hyperparameters used in OVOGO.

Component	Hyperparameter	Value
RARC	Ensemble members M	5
	Q heads per member	2
	Total scalar Q heads	10
	Value support atoms C	201
	Value support range	$[-0.1, 1.1]$
	Discount factor γ	0.995
	τ_{base}	0.7
	Entropy coefficient α_{ent}	0.2
	Minimum quantile $\tau_{\min}$	0.5
	Maximum quantile $\tau_{\max}$	0.7
	Target EMA η_{tgt}	0.005
	Value-loss weight λ_V	1.0
	Ranking-loss weight λ_R	1.0
	Optimizer	AdamW
	Learning rate	1×10^{-4}
	Weight decay	1×10^{-4}
	Global batch size	32
	Gradient-norm clipping	5
Ranking	Perturbation scale δ	0.04
	Ranking temperature τ_R	0.1
	Maximum gap $g_{\max}$	1.0
Actor	Optimizer	Adafactor
	Learning rate	1×10^{-4}
	State batch size	8
	Candidates per state G	8
	Candidate microbatch size	16
	Flow transitions K	10
	PPO clipping ε	0.01
	Log-ratio clipping c_ρ	20
	Gradient-norm clipping	1
Retention	Nominal FM coefficient λ_0	0.005
	Gradient fraction ρ_{BC}	0.2
	Numerical constant ϵ_g	10^{-12}

B.4 Baseline Configurations

All policy-training baselines start from the same pretrained policy and use the same initial offline dataset as OVOGO.

Success BC. Success BC performs supervised adaptation using successful trajectories from the same offline replay dataset. We optimize the native flow-matching objective of $\pi_{0.5}$ while keeping the pretrained vision-language backbone frozen and updating the action expert. Failed replay trajectories do not contribute to the Success BC objective.

RECAP. We implement RECAP (Intelligence et al., 2025a) using the official Evo-RL implementation (Contributors, 2026). RECAP starts from the same pretrained $\pi_{0.5}$ policy and the same initial replay dataset as OVOGO. We follow the official Evo-RL training configuration for advantage-conditioned policy adaptation.

Best-of-N. Best-of-N is an inference-time selection baseline and does not update the policy parameters. At each observation, the policy generates $N = 8$ candidate action chunks. The candidates are evaluated using the same frozen RARC used by OVOGO, and the candidate with the highest aggregated critic score is executed. Using the same candidate budget as OVOGO isolates the benefit of policy optimization from the benefit of critic-based candidate selection.

QGF. QGF is an inference-time value-guidance baseline. We use the same pretrained policy and the same RARC critic as OVOGO. During flow generation, action gradients of the critic are used to steer the generation trajectory toward higher-value actions, while the actor parameters themselves remain unchanged. This comparison separates inference-time value guidance from parameter-space policy optimization.

B.5 Offline Data and Training Protocol

We distinguish the standard offline setting used for the main comparison from the iterative offline experiment conducted on RoboTwin.

Standard offline setting. For the main comparison, all methods are trained from the same fixed offline dataset and receive no additional environment interactions during optimization.

RoboTwin contains ten tasks. For each task, the initial replay dataset contains 100 policy rollout episodes and 50 demonstration episodes. The complete initial RoboTwin dataset therefore contains 1000 policy rollout episodes and 500 demonstration episodes.

DexJoCo contains eleven tasks. Each task likewise contains 100 policy rollout episodes and 50 demonstration episodes, resulting in 1100 policy rollout episodes and 550 demonstration episodes in total.

Within the standard offline setting, the replay dataset remains fixed throughout critic learning and actor optimization. Fresh candidate actions generated by the flow policy are used only to construct policy-improvement signals. They are not treated as new environment transitions, are not assigned synthetic rewards, and are not inserted into the replay dataset.

Iterative offline setting on RoboTwin. We additionally evaluate OVOGO and RECAP under an iterative offline protocol on RoboTwin. Let $\mathcal{D}^{(k)}$ denote the replay dataset available in round k. Critic and actor optimization within that round use only the fixed dataset $\mathcal{D}^{(k)}$.

After completing the offline optimization stage, the updated policy is deployed to collect 100 new rollout episodes for each of the ten RoboTwin tasks, giving 1000 new rollout episodes per round. We denote this newly collected experience by $\mathcal{E}^{(k+1)}$ and construct the replay dataset for the following round as

$$\mathcal{D}^{(k+1)} = \mathcal{D}^{(k)} \cup \mathcal{E}^{(k+1)}. \tag{31}$$

The original demonstration trajectories remain available throughout all rounds.

This protocol preserves the offline nature of each optimization round: new physical experience is collected only between rounds, whereas policy-generated candidates sampled during optimization never enter the replay dataset.

B.6 Evaluation Protocol

For every method, we evaluate each task independently using 100 evaluation episodes. For task j, the episode success rate is

$$\mathrm{SR}_j = \frac{N_j^{\mathrm{succ}}}{N_j^{\mathrm{eval}}}, \tag{32}$$

where N_j^{succ} and N_j^{eval} denote the number of successful and total evaluation episodes for task j, respectively.

The benchmark-level metric reported in the main paper is the task-macro success rate. For a benchmark containing J tasks,

$$\mathrm{MacroSR} = \frac{1}{J}\sum_{j=1}^{J} \mathrm{SR}_j\,. \tag{33}$$

Thus, every task contributes equally to the final score regardless of differences in its individual success rate.

Table 5: Per-task success rates (%) on the ten RoboTwin tasks after one offline training round. The last row reports the task-macro average.

Task	$\pi_{0.5}$	Success BC	RECAP	Best-of-N	QGF	**OVOGO**
adjust_bottle	70.0	58.0	51.0	74.0	69.0	74.0
click_alarmclock	65.0	76.0	62.0	86.0	72.0	70.0
click_bell	52.0	84.0	78.0	62.0	56.0	58.0
dump_bin_bigbin	70.0	44.0	53.0	76.0	71.0	74.0
handover_mic	66.0	38.0	24.0	72.0	70.0	86.0
open_laptop	45.0	46.0	46.0	34.0	45.0	52.0
pick_dual_bottles	61.0	60.0	14.0	60.0	58.0	70.0
place_a2b_right	39.0	18.0	22.0	30.0	42.0	42.0
place_object_stand	77.0	46.0	28.0	80.0	78.0	86.0
press_stapler	62.0	68.0	73.0	72.0	62.0	66.0
Task-macro average	60.7	53.8	45.1	64.6	62.3	**67.8**

Table 6: Per-task success rates (%) on the eleven DexJoCo tasks after one offline training round. The last row reports the task-macro average.

Task	$\pi_{0.5}$	Success BC	RECAP	Best-of-N	QGF	**OVOGO**
bimanual_assembly	0.0	0.0	0.0	0.0	0.0	0.0
bimanual_hanoi	2.0	2.0	0.0	2.0	2.0	4.0
bimanual_microwave_cook	28.0	26.0	26.0	32.0	30.0	42.0
bimanual_photograph	9.0	7.0	1.0	9.0	8.0	17.0
bimanual_unlock_ipad	0.0	0.0	0.0	0.0	0.0	0.0
click_mouse	50.0	69.0	67.0	53.0	53.0	61.0
fold_glasses	31.0	27.0	26.0	35.0	34.0	45.0
hammer_nail	51.0	46.0	46.0	56.0	49.0	59.0
pick_bucket	52.0	43.0	21.0	49.0	55.0	57.0
pinch_tongs	32.0	4.0	7.0	39.0	37.0	50.0
water_plant	48.0	58.0	52.0	51.0	57.0	59.0
Task-macro average	27.5	25.6	22.4	29.6	29.5	**35.8**

All compared methods within the same benchmark use the same task split, language instructions, observation processing, action normalization, policy initialization, and evaluation protocol. RoboTwin policies generate 50-step action chunks, while DexJoCo policies generate 30-step action chunks.

For Best-of-N, the only additional inference-time computation is the generation and critic evaluation of $N = 8$ candidates. For QGF, the critic is used only for inference-time gradient guidance. For OVOGO, evaluation uses the optimized policy directly without critic-based candidate selection.

Per-task results. Tables 5 and 6 report the corresponding per-task success rates for RoboTwin and DexJoCo under the standard offline setting.

B.7 Compute Platform

All experiments are implemented in PyTorch and trained on NVIDIA A100 GPUs. We use four GPUs for distributed critic and actor training.

During critic optimization, the Gemma 3 270M language backbone and the SigLIP2-So400M vision backbone with 224×224 inputs remain frozen. Only the critic-specific projection and prediction modules are updated. During actor optimization, RARC and the pretrained vision-language backbone remain frozen, and only the $\pi_{0.5}$ action expert is optimized.

The benchmark-specific policy checkpoints and the DexJoCo JAX/Orbax-to-PyTorch conversion are described in Section B.1; we do not repeat the checkpoint URLs here.