FLOW-KTO: LEARNING FLOW POLICIES FROM DEPLOYMENT FAILURES FOR REAL-WORLD ROBOT MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In real-world robot manipulation, flow policies can perform complex tasks after supervised fine-tuning (SFT), yet still make recurring errors during deployment. Expanding the expert demonstration dataset to correct these errors can be costly (Wagenmaker et al., 2025). Deployment failures provide another source of information for policy refinement. For flow policies, however, direct optimization through action generation can produce unstable gradients, and action log-probabilities are costly to evaluate (Park et al., 2025; Skreta et al., 2025). We propose Flow-KTO, an offline post-training method for flow policies. It learns from success and failure feedback on recorded execution clips. Flow-KTO accounts for asynchronous inference latency to align execution feedback with the corresponding action trajectories. The method replaces direct computation of policy log-likelihood ratios with a surrogate, avoiding backpropagation through action generation. A policy drift estimate modulates the strength of feedback updates at each training step. Across four real-robot tasks, Flow-KTO reduces the average target-mistake recurrence rate by 33.3% relative to the strongest baseline, while improving average success rate by 13.3% and increasing task throughput by 17.7% on average.

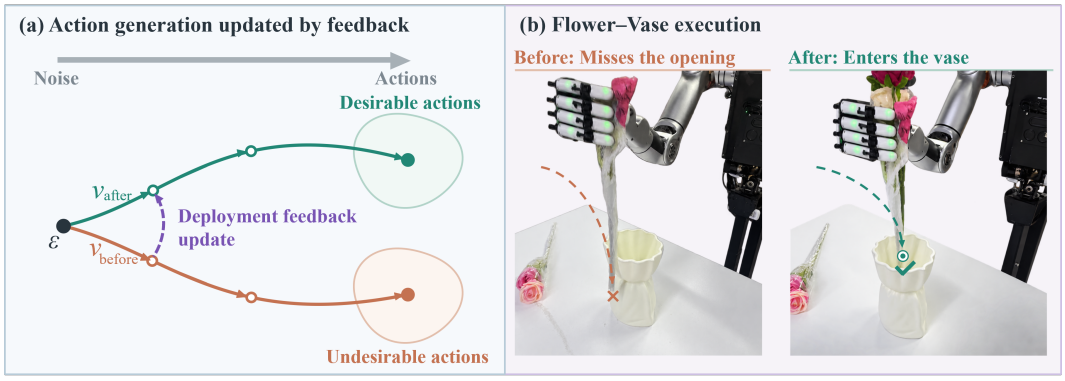

Figure 1: **Updating flow policies with deployment feedback.** (a) Conceptual generation paths before and after offline feedback post-training, from a noise sample ϵ to undesirable or desirable actions. The purple arrow denotes a change in the learned flow after training. (b) Flower–Vase photographs illustrate unsuccessful and successful insertion. The paths and before/after comparison are schematic, not a controlled comparison of policy checkpoints.

1 INTRODUCTION

Recently, flow policies have become a prominent approach to robot manipulation (Black et al., 2024; Bjorck et al., 2025). Large-scale imitation learning has enabled these policies to perform challenging manipulation tasks, yet they continue to make mistakes when deployed in the physical world. A

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.

common approach to addressing these errors is to refine the policy through imitation learning on additional expert demonstrations with broader state coverage (Ross et al., 2011; Liu et al., 2023). Collecting such data can be costly on real robots (Wagenmaker et al., 2025). An alternative is to learn directly from deployment failures, reducing reliance on additional expert demonstrations (Hao et al., 2026). However, using these failures to improve flow policies remains challenging.

Many post-training methods (Wallace et al., 2024; Liang et al., 2025; Liu et al., 2025b;a; Xue et al., 2025; Wu et al., 2026) improve the policy by comparing better and worse samples under the same conditioning context. In real-world robot manipulation, however, executing an action may change the environment. Evaluating alternative actions under the same conditions can therefore require resetting the scene, which is costly and difficult to do precisely. Designing detailed rewards for real-world manipulation can also require substantial task-specific effort (Luo et al., 2024b; Dong et al., 2026). Alternatively, deployment feedback can be obtained by labeling each recorded execution clip without requiring paired comparisons. Failed clips provide negative feedback for policy refinement, while successful clips provide positive feedback. Labels can be provided by human annotators or visual success classifiers (Luo et al., 2024b). Learned value models provide another source of feedback (Physical Intelligence et al., 2025; Huang et al., 2026). Recent work has also used independently labeled samples to improve language models (Ethayarajh et al., 2024; Jung et al., 2025), diffusion models for image generation (Li et al., 2024), and visuomotor diffusion policies (Moletta et al., 2026).

Optimizing flow policies with deployment feedback presents additional challenges. Policy gradient methods such as PPO and GRPO rely on policy log-probabilities (Schulman et al., 2017; Ouyang et al., 2022; Shao et al., 2024), which are readily available for Gaussian control policies and autoregressive language models. These likelihood-based methods are difficult to apply directly to flow policies because action log-probabilities are costly to evaluate (Skreta et al., 2025). An alternative is to score generated actions with a differentiable reward and propagate its gradient back to the policy parameters (Clark et al., 2024). Although this avoids likelihood evaluation, the gradient must pass through the action-generation process, which in a flow policy involves numerically solving an ordinary differential equation (ODE). Unrolling the ODE solver for backpropagation through time (BPTT) creates a computation graph spanning all integration steps, increasing memory usage and potentially destabilizing optimization (Chen et al., 2018; Park et al., 2025). Recent methods use a critic's action gradients for policy improvement without full sampler BPTT (Li & Levine, 2026; Zhou et al., 2026; Dong et al., 2026; Wang et al., 2026b). In real-world robot manipulation, learning $Q(s, a)$ requires estimating the expected return of an action from limited interaction data (Luo et al., 2024a). Policy improvement further depends on reliable action gradients; errors in these gradients can misdirect optimization even when the critic fits observed returns well (D'Oro & Jaśkowski, 2020).

Addressing these challenges, we propose Flow-KTO, an offline post-training method for flow policies. The method aims to reduce recurring errors using deployment feedback from recorded execution clips. We align clip-level feedback with training action trajectories by accounting for the temporal offset between asynchronous policy inference and robot execution. Figure 1 illustrates the intended effect on action generation alongside a Flower–Vase example. We use the difference in ELBO-weighted conditional flow-matching (CFM) losses between the trainable policy and a frozen pretrained reference as a surrogate for the policy log-likelihood ratio, avoiding backpropagation through action generation. A bounded objective encourages better fit to trajectories from successful clips and reduced fit to those from failed clips. A policy drift estimate modulates the strength of feedback updates at each training step. The objective also includes behavior-cloning regularization to help retain previously learned behaviors.

2 Related Work

Offline policy learning for robot manipulation. Offline policy learning trains robot policies from previously collected data without additional environment interaction during optimization. Behavior cloning learns to reproduce demonstrated actions through supervised learning (Chi et al., 2023; Black et al., 2024). For data of varying quality, filtering selects which samples to imitate, while reward- or advantage-weighted objectives adjust their contribution to training (Peng et al., 2019; Kostrikov et al., 2022; Mao et al., 2026). Conditioned policies can also learn from such data by

taking returns or advantage labels as additional inputs during offline training. At inference, return-conditioned policies use a high target return as input. Advantage-conditioned policies use a label indicating improvement over the reference policy (Kumar et al., 2019; Physical Intelligence et al., 2025).

Value learning uses the previously collected data to estimate expected future returns. These estimates can provide advantages for weighted or conditioned policy training (Kostrikov et al., 2022; Physical Intelligence et al., 2025), or guide updates toward actions with higher predicted returns (Wang et al., 2023; Fang et al., 2025). The learned value function may overestimate returns for actions poorly represented in the dataset. Policy updates can then favor these actions because of estimation errors. To reduce this risk, methods keep the policy close to the behavior in the dataset or lower value estimates when uncertainty is high (Wang et al., 2023; Fang et al., 2025; Gao et al., 2025).

Optimizing diffusion and flow policies. Diffusion and flow policies can be improved by updating their parameters during training or by changing how actions are generated at inference. During training, direct value maximization differentiates a critic objective through the denoising process (Wang et al., 2023). To avoid differentiating through the full sampler, other methods approximate denoised actions or optimize a distilled one-step actor (Kang et al., 2023; Park et al., 2025). Value information can also be converted into denoising or flow-matching objectives (Psenka et al., 2024; Fang et al., 2025; Zhang et al., 2025a). Adjoint matching provides velocity supervision from critic gradients without full sampler BPTT (Li & Levine, 2026). Policy-gradient methods use likelihoods of stochastic denoising transitions (Ren et al., 2025; Zhang et al., 2025b) or CFM-based likelihood-ratio surrogates in advantage-weighted updates (McAllister et al., 2025; Yi et al., 2026). Reward-derived positive and negative signals can also train flow models directly through forward-process regression (Zheng et al., 2026).

At inference, learned values can select among sampled actions (Hansen-Estruch et al., 2023; Nakamoto et al., 2024), or critic gradients can guide the flow sampler (Zhou et al., 2026). Classifier-free guidance can also steer action generation without an explicit value function (Frans et al., 2025).

Learning from positive and negative feedback. Generative models can be aligned using paired comparisons or independent labels. Pairwise objectives compare outputs under a shared condition and have been adapted to flow-based VLAs using CFM loss surrogates (Rafailov et al., 2023; Liu, 2026; Wu et al., 2026). An alternative assigns positive or negative feedback to each sample independently. KTO learns from these labels through a bounded utility, whereas BCO uses a binary classification loss (Ethayarajh et al., 2024; Jung et al., 2025). For diffusion models, denoising-based likelihood surrogates enable learning from positive and negative feedback in image generation (Li et al., 2024). ELBO-KTO applies a bounded objective to masked diffusion language models using ELBO differences from a reference model (Jindal et al., 2025). In robotics, utility-based objectives refine autoregressive action policies using intervention-derived labels (Xia et al., 2025), while visuomotor diffusion policies use observation similarities to weight utilities across demonstrations with positive and negative feedback (Moletta et al., 2026). Behavior-mode labels provide another route: a mode classifier supplies guidance that is subsequently distilled into policy weights (Wang et al., 2026a). Flow-KTO refines continuous flow policies using deployment feedback. Its bounded objective uses ELBO-weighted CFM loss differences relative to a frozen pretrained policy as a likelihood-ratio surrogate.

3 Problem Setup and Preliminaries

3.1 Problem Setup

The goal is to refine a pretrained flow policy offline using deployment feedback on recorded execution clips. Failed clips provide negative feedback, while successful clips provide positive feedback. Feedback can be provided by human annotators or obtained from visual success classifiers (Luo et al., 2024b) and learned value models (Physical Intelligence et al., 2025; Huang et al., 2026).

Let o denote the observations and task conditioning, and let $a \in \mathbb{R}^{H \times d}$ denote an action trajectory with H time steps and d action coordinates per step. Each feedback annotation assigns positive or negative feedback to a variable-length execution interval $[\tau, \tau + n]$. For a training trajectory starting

at frame i, let s_i denote the label inherited from its associated annotation:

$$s_i = \begin{cases} +1, & \text{positive feedback,} \\ -1, & \text{negative feedback.} \end{cases} \tag{1}$$

Each annotation is evaluated independently, without paired comparisons. Training uses a fixed dataset of labeled action trajectories,

$$\mathcal{D} = \{(o_i, a_i, s_i)\}, \tag{2}$$

where o_i is the observation and task conditioning associated with trajectory a_i.

3.2 Flow Policies and Flow Matching

A flow policy maps Gaussian noise to an action trajectory using a velocity field conditioned on o (Lipman et al., 2023; Li & Levine, 2026). The velocity field v_θ is represented by a neural network with parameters θ. For a training trajectory a, a standard flow-matching path interpolates between noise and data to construct a_t:

$$a_t = (1-t)\epsilon + ta, \qquad \epsilon \sim \mathcal{N}(0, I), \quad t \in [0, 1], \tag{3}$$

where t is flow time, distinct from the physical time steps in the trajectory. To generate an action trajectory, the policy starts from noise and integrates the learned velocity field. The evolving sample a_t^{gen} follows the ordinary differential equation (ODE)

$$\frac{\mathrm{d}a_t^{\text{gen}}}{\mathrm{d}t} = v_\theta(a_t^{\text{gen}}, t \mid o), \qquad a_0^{\text{gen}} = \epsilon. \tag{4}$$

The endpoint a_1^{gen} is a sample from $\pi_\theta(\cdot \mid o)$, the resulting conditional distribution of action trajectories.

For this linear path, conditional flow matching (CFM) trains the velocity field to predict the constant path velocity $a - \epsilon$. For a given trajectory and a sampled flow time and noise, the squared prediction error is

$$\ell_\theta = \|v_\theta(a_t, t \mid o) - (a - \epsilon)\|_2^2. \tag{5}$$

With flow time sampled uniformly, the CFM loss averages this error over time and noise:

$$L_\theta(o, a) = \mathbb{E}_{t,\epsilon}[\ell_\theta]. \tag{6}$$

4 Method

Flow-KTO uses deployment feedback as direct supervision for offline policy refinement, aiming to reduce the recurrence of observed errors during real-world robot deployment. To avoid costly action log-probability evaluation and backpropagation through action generation, we use a likelihood-ratio surrogate based on CFM. To help preserve pretrained capabilities during refinement, we combine a behavior-cloning (BC) anchor with policy drift estimation, which modulates the strength of feedback updates.

Flow-KTO initializes the trainable policy π_θ from a pretrained policy π_{ref}, whose parameters θ_{ref} remain frozen.

4.1 Likelihood-Ratio Surrogate

The evidence lower bound (ELBO) connects CFM loss to a lower bound on action log-likelihood. For the linear path in Eq. (3), this connection uses a denoising-time-weighted version of the CFM loss (Kingma & Gao, 2023). To avoid the singular endpoint, we approximate the full time interval by sampling $t \sim \mathcal{U}(0, 1-\delta)$, where $\delta \in (0, 1)$. All ELBO-related quantities below use this truncated interval; we suppress their dependence on δ for readability:

$$\begin{aligned} L_\theta^{\text{ELBO}}(o, a) &= (1-\delta)\mathbb{E}_{t,\epsilon}\left[\frac{t}{1-t}\,\ell_\theta\right] \\ &= -\,\text{ELBO}_\theta(a \mid o) + c. \end{aligned} \tag{7}$$

The cutoff bounds the time weight by $(1-\delta)/\delta$, removing the endpoint singularity. The factor $1-\delta$ preserves the scale of the truncated time integral, and c is independent of the policy parameters for policies using the same forward process and cutoff. Within this derivation, π_θ denotes the likelihood under the truncated construction. The corresponding ELBO satisfies

$$\mathrm{ELBO}_\theta(a \mid o) = \log \pi_\theta(a \mid o) - D_\theta, \tag{8}$$

where $D_\theta \geq 0$ is the variational gap. Combining the current and reference identities cancels c and gives

$$\log \frac{\pi_\theta(a \mid o)}{\pi_{\mathrm{ref}}(a \mid o)} = L_{\mathrm{ref}}^{\mathrm{ELBO}}(o, a) - L_\theta^{\mathrm{ELBO}}(o, a) + D_\theta - D_{\mathrm{ref}}. \tag{9}$$

The loss difference is therefore a computable surrogate for the deployed policy's log-likelihood ratio; the cutoff and variational-gap difference are the two approximation terms. We use this difference as the reference-relative CFM score (McAllister et al., 2025):

$$g_\theta(o, a) = L_{\mathrm{ref}}^{\mathrm{ELBO}}(o, a) - L_\theta^{\mathrm{ELBO}}(o, a). \tag{10}$$

In practice, the current and reference losses use the same sampled flow times and noise.

4.2 Current-Policy Drift Estimation

To retain the capabilities acquired during pretraining and promote stable policy improvement, we aim to avoid excessive drift from the pretrained policy during feedback optimization.

We quantify policy drift with KL divergence, which measures the difference between the current and pretrained policies' action distributions under the same context. We express KL as the expectation of the log-likelihood ratio, which can be estimated using the CFM surrogate:

$$\begin{aligned} D_{\mathrm{KL}}(\pi_\theta(\cdot \mid o) \,\|\, \pi_{\mathrm{ref}}(\cdot \mid o)) &= \int \pi_\theta(\widetilde{a} \mid o) \log \frac{\pi_\theta(\widetilde{a} \mid o)}{\pi_{\mathrm{ref}}(\widetilde{a} \mid o)} \, \mathrm{d}\widetilde{a} \\ &= \mathbb{E}_{\widetilde{a} \sim \pi_\theta(\cdot \mid o)} \left[\log \frac{\pi_\theta(\widetilde{a} \mid o)}{\pi_{\mathrm{ref}}(\widetilde{a} \mid o)} \right] \\ &\approx \mathbb{E}_{\widetilde{a} \sim \pi_\theta(\cdot \mid o)} \left[g_\theta(o, \widetilde{a}) \right], \end{aligned} \tag{11}$$

For a minibatch of B stored contexts, we sample one action trajectory per context from the current policy and average the corresponding CFM surrogate scores:

$$z = \frac{1}{B} \sum_{j=1}^{B} g_\theta(o_j, \widetilde{a}_j) = \frac{1}{B} \sum_{j=1}^{B} \left[L_{\mathrm{ref}}^{\mathrm{ELBO}}(o_j, \widetilde{a}_j) - L_\theta^{\mathrm{ELBO}}(o_j, \widetilde{a}_j) \right], \qquad \widetilde{a}_j \sim \pi_\theta(\cdot \mid o_j). \tag{12}$$

We update z during training using new action trajectories generated by the current policy. The feedback objective compares each labeled trajectory's score with z to adjust the strength of its feedback update. We treat z as a constant when computing the feedback gradients, avoiding backpropagation through action generation.

4.3 Training Objective

Latency-aware feedback alignment. Policy inference and robot execution proceed asynchronously, producing a fixed offset of k control steps between a generated action chunk and the corresponding observed behavior. We account for this latency when assigning an annotation over $[\tau, \tau + n]$ to H-step training trajectories. The admissible start indices are

$$\mathcal{R} = [\tau - k, \min\{\tau + n - H,\ \tau + n - k\}]. \tag{13}$$

Each $i \in \mathcal{R}$ contributes a sample (o_i, a_i, s_i), where a_i is the H-step action chunk beginning at frame i and s_i is the annotation's label. Applying this construction to all annotations gives the dataset in Eq. (2). Figure 2 illustrates the latency shift and the exclusion of starts whose H-step chunks extend beyond the annotation end.

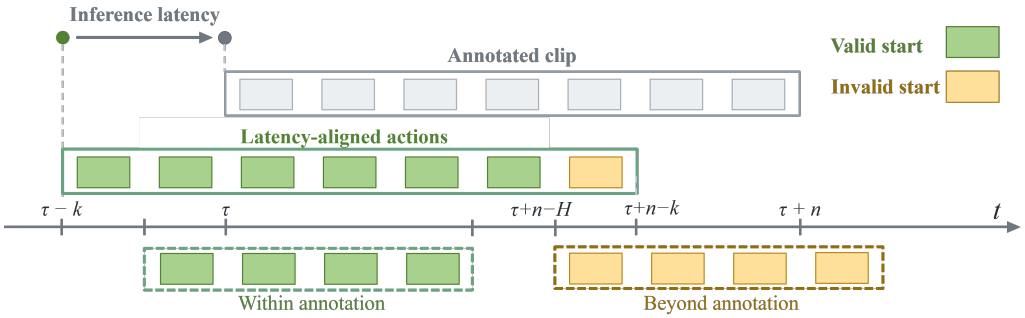

Figure 2: **Latency-aware feedback alignment.** An annotation over $[\tau, \tau + n]$ is shifted back by k control steps to align feedback with generated actions. Green squares denote valid training starts. Yellow squares denote starts excluded because their H-step chunks extend beyond the annotation end. The lower row contrasts valid and invalid training chunks.

Feedback-weighted CFM updates. Applying the bounded KTO objective (Ethayarajh et al., 2024) to the centered reference-relative score gives the weighted feedback term

$$\mathcal{L}_i^{\mathrm{fb}} = \lambda_{s_i}\, \sigma\big(-s_i \beta \big[g_\theta(o_i, a_i) - z\big]\big)\,, \tag{14}$$

where σ is the logistic sigmoid, $\beta > 0$ controls its sensitivity to the centered score, and λ_{s_i} selects $\lambda_+ > 0$ or $\lambda_- > 0$ according to the feedback label. With π_{ref} fixed and gradients stopped through z, the trainable dependence of $\mathcal{L}_i^{\mathrm{fb}}$ enters through the ELBO-weighted CFM loss. Differentiation gives

$$\nabla_\theta \mathcal{L}_i^{\mathrm{fb}} = \lambda_{s_i} s_i \beta\, \sigma'\big(\beta\big[g_\theta(o_i, a_i) - z\big]\big)\, \nabla_\theta L_\theta^{\mathrm{ELBO}}(o_i, a_i), \tag{15}$$

Minimizing $\mathcal{L}_i^{\mathrm{fb}}$ follows gradient descent on the ELBO-weighted CFM loss for positive feedback and gradient ascent for negative feedback. The nonnegative sigmoid derivative scales this update according to the current centered score. The CFM input for the fixed trajectory satisfies

$$a_t = (1 - t)\epsilon + t a_i, \qquad \frac{\partial a_t}{\partial \theta} = 0. \tag{16}$$

The loss $L_\theta^{\mathrm{ELBO}}(o_i, a_i)$ depends on θ only through the velocity network. Computing its gradient does not require differentiating through the ODE used to generate policy actions.

Overall objective. We combine the feedback terms with a BC anchor to preserve pretrained behavior during policy updates. Let $\mathcal{D}_+$ and $\mathcal{D}_-$ denote the subsets of $\mathcal{D}$ with positive and negative feedback, respectively, and let $\mathcal{D}_{\mathrm{BC}}$ denote a fixed BC dataset. Averaging the feedback terms separately over the two label classes gives

$$\begin{aligned} \mathcal{L}(\theta) = {} & \lambda_+ \mathbb{E}_{\mathcal{D}_+}\big[\sigma\big(\beta\big[z - g_\theta(o, a)\big]\big)\big] \\ & + \lambda_- \mathbb{E}_{\mathcal{D}_-}\big[\sigma\big(\beta\big[g_\theta(o, a) - z\big]\big)\big] \\ & + \lambda_{\mathrm{BC}} \mathbb{E}_{\mathcal{D}_{\mathrm{BC}}}[L_\theta(o, a)], \end{aligned} \tag{17}$$

where $\lambda_{\mathrm{BC}} \geq 0$ controls the BC anchor. The BC term uses the standard CFM loss.

5 Experiments

We evaluate Flow-KTO on four real-robot tasks. We first assess whether the complete method improves task completion and reduces target-mistake recurrence relative to the initial policy and prior methods, with throughput as a complementary measure of execution benefit. We then use four ablations to examine the roles of negative feedback, latency-aware feedback alignment, drift estimation, and the BC anchor.

5.1 Experimental Setup

Evaluation tasks. The four tasks in Figure 3 cover flower–vase insertion, picking and scanning six items, bagging those items, and scooping popcorn into a bowl: precise insertion, repeated object handling, packing, and tool use.



(a) Flower–Vase Insertion Align and insert the stem

(b) Pick-and-Scan Pick, scan, and stage each item

(c) Bagging Place all scanned items in the bag

(d) Popcorn Scooping Scoop 6–10 g into the bowl

Figure 3: **Real-robot evaluation tasks.** Representative scenes from the four tasks used in our experiments.

Feedback and evaluation. All feedback comes from autonomous execution of the frozen initial policy. Seedance 2.0 Pro labels each executed action chunk as desirable or undesirable; no expert correction replaces what the robot did. All policies in the main comparison are evaluated on the same fixed 25 trials per task, including 10 layouts drawn from initial-policy failures. Label auditing and evaluation procedures are in Appendix A.

Evaluation metrics. We report success rate (SR), target-mistake recurrence rate (TMR), and throughput. Each trial is one task attempt; success requires the complete task rubric, and TMR records whether a predefined target mistake occurs:

$$\begin{aligned} \mathrm{SR} &= \frac{\text{successful task attempts}}{\text{total task attempts}} \times 100\%, \\ \mathrm{TMR} &= \frac{\text{task attempts with a target mistake}}{\text{total task attempts}} \times 100\%. \end{aligned} \tag{18}$$

Both rates include all attempts; TMR counts each at most once, including repeated mistakes or subsequent recovery. Throughput measures successful task executions per hour. Task rubrics, mistake annotation, and timing conventions are given in Appendix A.2.

5.2 Comparison with Prior Methods

We compare Flow-KTO with the frozen initial policy, supervised fine-tuning (SFT), a fixed-data variant of FPO (McAllister et al., 2025), and conditioned behavior cloning (conditioned BC) (Physical Intelligence et al., 2025). All trained methods start from the same pretrained initial policy and draw deployment data from its frozen real-robot rollouts. SFT applies standard CFM to desirable chunks and a fixed BC pool. FPO uses desirable and undesirable labels as signed advantage proxies in its clipped flow-policy objective; conditioned BC uses these labels as binary conditions and selects the desirable condition at evaluation. All policies follow the same task protocol, with details in Appendix A.1.

Comparison results. Flow-KTO improves task completion on all four tasks (Table 1). Its average SR is 68.0%, compared with 34.0% for the initial policy, 45.0% for SFT, 51.0% for conditioned BC, and 60.0% for FPO: gains of 34, 23, 17, and 8 percentage points, respectively. The gain over FPO ranges from 4 to 12 percentage points across tasks.

Flow-KTO also reduces average TMR to 20.0%, compared with 56.0% for the initial policy, 41.0% for SFT, 39.0% for conditioned BC, and 30.0% for FPO. Successful-task throughput exceeds FPO on every task, including 96 versus 83.6 tasks/h on Flower–Vase Insertion and 20 versus 15.4 tasks/h on Bagging. The gains in task completion thus coincide with fewer target mistakes and more successful executions per hour. Initial-policy failures and corresponding Flow-KTO executions at two decision points are shown in Appendix Figure 5.

Table 1: **Post-training comparison.** Each method is evaluated by success rate (SR, %), target-mistake recurrence rate (TMR, %), and throughput (successful tasks/h). Avg. gives the four-task macro average for SR and TMR; throughput is reported per task. Bold marks the best macro averages; a dash denotes an inapplicable aggregate.

Method	Metric	Flower–Vase Insertion	Pick-and-Scan	Bagging	Popcorn Scooping	Avg.
Initial	SR ↑	52	32	24	28	34.0
	TMR ↓	44	60	64	56	56.0
	Throughput ↑	52	8	7	14	—
SFT	SR ↑	60	44	36	40	45.0
	TMR ↓	32	48	52	32	41.0
	Throughput ↑	70	12	11	22	—
Conditioned BC	SR ↑	64	52	44	44	51.0
	TMR ↓	32	44	52	28	39.0
	Throughput ↑	75	14	12.5	24	—
FPO	SR ↑	72	64	48	56	60.0
	TMR ↓	24	36	32	28	30.0
	Throughput ↑	83.6	20	15.4	33	—
Flow-KTO (ours)	SR ↑	80	68	60	64	**68.0**
	TMR ↓	12	24	28	16	**20.0**
	Throughput ↑	96	21	20	40	—

5.3 Ablation Studies

We ablate negative feedback, latency-aware feedback alignment, drift estimation, and the BC anchor on Flower–Vase Insertion and Popcorn Scooping. Variants share initialization, frozen reference, source rollouts and labels, optimization settings, and held-out layouts. Removing latency alignment changes the training chunks selected from those annotations; the other ablations change an objective component or training duration. Each task–variant pair uses one training run, so comparisons are descriptive. Figure 4 shows TMR for negative feedback and latency alignment, and SR for latency alignment, drift estimation, and the BC anchor. Detailed results appear in Appendix A.4.

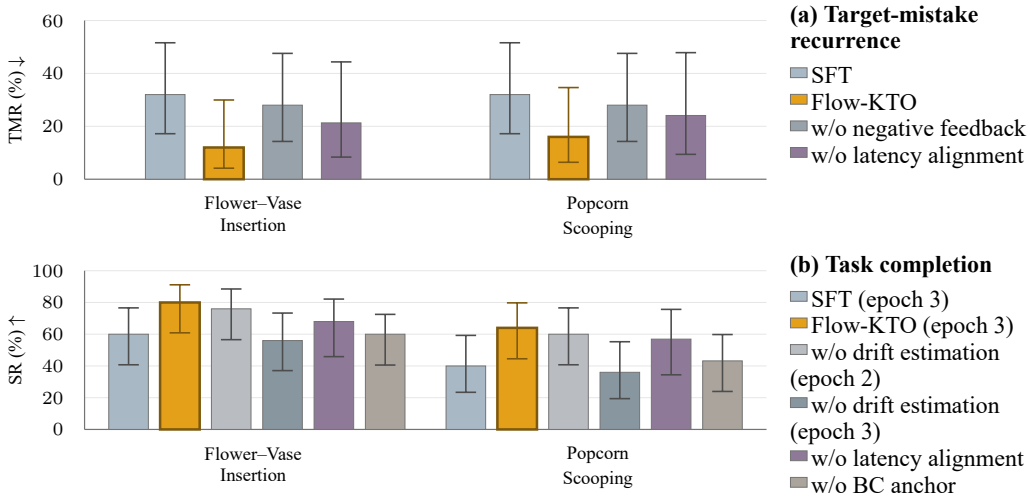

Figure 4: **Component ablations.** TMR (a) and SR (b) for Flower–Vase Insertion (left group) and Popcorn Scooping (right group). Whiskers show fluctuation ranges for the variants without latency alignment or the BC anchor, and for the Popcorn Scooping variant without drift estimation (epoch 3); other whiskers are 95% Wilson intervals over 25 trials. Detailed rate and throughput plots appear in Appendix A.4.

Negative feedback. Figure 4(a) compares SFT, full Flow-KTO, and a variant that sets $\lambda_- = 0$ while retaining the remaining Flow-KTO objective. SFT provides a standard positive-only baseline, and the $\lambda_- = 0$ variant measures the effect of removing negative feedback within Flow-KTO. All three configurations use the same training duration. A norm-matched control and throughput plot appear in Appendix A.4.

Negative feedback reduces target-mistake recurrence and improves successful task throughput on both tasks. Setting $\lambda_- = 0$ raises TMR from 12% to 28% on Flower–Vase Insertion and from 16% to 28% on Popcorn Scooping. Throughput decreases from 96 to 74 and from 40 to 25 successful tasks/h, respectively. With the negative term removed, TMR is only 4 percentage points lower than SFT on either task. These comparisons within the same objective indicate that explicit negative feedback helps suppress recurring mistakes beyond positive-only training.

Drift estimation. The drift term centers feedback updates (Eq. 15). Figure 4(b) compares SFT and full Flow-KTO at epoch 3 with a variant without drift estimation ($z = 0$) at epochs 2 and 3, measuring how continued optimization affects SR and throughput on each task.

Without drift estimation, performance deteriorates between epochs 2 and 3. SR falls from 76% to 56% on Flower–Vase Insertion and from 60% to 36% on Popcorn Scooping, decreases of 20 and 24 percentage points, respectively. Throughput falls from 90 to 52 and from 37 to 18 successful tasks/h, respectively.

At epoch 3, full Flow-KTO reaches 80% and 64% SR on the two tasks, 24 and 28 percentage points above the variant without drift estimation, respectively. Its throughput is also higher, at 96 versus 52 and 40 versus 18 successful tasks/h. The checkpoint without drift estimation at epoch 2 is only 4 percentage points below full Flow-KTO at epoch 3 in SR on each task, making its subsequent decline especially relevant. These results suggest that drift estimation helps maintain task performance during continued feedback optimization under the current training recipe.

Latency-aware feedback alignment. Setting $k = 0$ in Eq. 13 removes the latency offset while retaining complete H-step chunks, the same source rollouts and labels, feedback objective, BC pool, and training schedule. SR falls from 80% to 68% on Flower–Vase Insertion and from 64% to 56.9% on Popcorn Scooping; TMR rises from 12% to 21.3% and from 16% to 24.1%, respectively. On both tasks, TMR remains below the 28% observed without negative feedback. These results suggest that alignment helps task completion and reduces target mistakes under the current training recipe.

BC anchor. Figure 4(b) also compares SFT, full Flow-KTO, and a variant that removes the BC anchor by setting $\lambda_{\mathrm{BC}} = 0$. The two Flow-KTO configurations share the remaining feedback objective and training duration.

Removing the anchor reduces SR from 80% to 60% on Flower–Vase Insertion and from 64% to 43.2% on Popcorn Scooping, decreases of 20 and 20.8 percentage points, respectively. Throughput decreases from 96 to 90 and from 40 to 36 successful tasks/h, respectively. These results suggest that the BC anchor supports task completion during feedback optimization on both tasks.

6 Conclusion

We presented Flow-KTO, an offline post-training method that refines flow policies using success and failure feedback from recorded execution clips. The method accounts for asynchronous inference latency when aligning feedback with action trajectories and uses a CFM-based surrogate for policy log-likelihood ratios, avoiding backpropagation through action generation. A policy drift estimate modulates feedback updates during training. Across four real-robot tasks, Flow-KTO achieves a macro-average success rate of 68.0% and a target-mistake recurrence rate of 20.0%, improving over the fixed-data FPO baseline by 8 and 10 percentage points, respectively. Ablations further show that negative feedback helps reduce recurring mistakes beyond positive-only training. These results highlight the value of deployment failures as explicit negative supervision for flow-policy refinement. Our evaluation focuses on a single round of offline post-training with fixed feedback datasets. Future work could examine whether iterative policy updates using newly collected deployment feedback sustain these improvements, and how feedback annotation errors affect policy refinement.

AI USE STATEMENT

We used AI tools for language polishing, literature retrieval and discovery, and assistance with code implementation for reproducing prior methods. The authors take responsibility for the final content and results.

REFERENCES

Johan Bjorck, Valts Blukis, Fernando Castañeda, et al. GR00T N1.5: An improved open foundation model for generalist humanoid robots. NVIDIA Research, June 2025. URL https://research.nvidia.com/labs/gear/gr00t-n1_5/.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://papers.nips.cc/paper/2018/hash/69386f6bb1dfed68692a24c8686939b9-Abstract.html.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023. URL https://arxiv.org/abs/2303.04137.

Kevin Clark, Paul Vicol, Kevin Swersky, and David J. Fleet. Directly fine-tuning diffusion models on differentiable rewards. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.17400.

Perry Dong, Kuo-Han Hung, Tian Gao, Dorsa Sadigh, and Chelsea Finn. EXPO-FT: Sample-efficient reinforcement learning finetuning for vision-language-action models. *arXiv preprint arXiv:2605.25477*, 2026. URL https://arxiv.org/abs/2605.25477.

Pierluca D'Oro and Wojciech Jaśkowski. How to learn a useful critic? model-based action-gradient-estimator policy optimization. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://papers.nips.cc/paper_files/paper/2020/hash/03255088ed63354a54e0e5ed957e9008-Abstract.html.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. KTO: Model alignment as prospect theoretic optimization. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2402.01306.

Linjiajie Fang, Ruoxue Liu, Jing Zhang, Wenjia Wang, and Bing-Yi Jing. Diffusion actor-critic: Formulating constrained policy iteration as diffusion noise regression for offline reinforcement learning. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ldVkAO09Km.

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

Chen-Xiao Gao, Chenyang Wu, Mingjun Cao, Chenjun Xiao, Yang Yu, and Zongzhang Zhang. Behavior-regularized diffusion policy optimization for offline reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 18630–18657. PMLR, 2025. URL https://proceedings.mlr.press/v267/gao25q.html.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. IDQL: Implicit Q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023. URL https://arxiv.org/abs/2304.10573.

Haoran Hao, Shahram Najam Syed, Jeffrey Ichnowski, and Jeff Schneider. FAR: Failure-aware retry for test-time recovery and continual policy improvement. *arXiv preprint arXiv:2607.01111*, 2026. URL https://arxiv.org/abs/2607.01111.

Dongchi Huang, Hongyin Zhang, Bohan Hou, Siteng Huang, Zhian Su, Hang Guo, Tong Lu, Zhaofeng Xu, Jiahao Tang, Jianfei Yang, Donglin Wang, Peixi Peng, Mingxiu Chen, Deli Zhao, and Xin Li. RynnValue: Scaling robotic value foundation models with temporal distance. *arXiv preprint arXiv:2608.09853*, 2026. URL https://arxiv.org/abs/2608.09853.

Vaibhav Jindal, Hejian Sang, Chun-Mao Lai, Yanning Chen, and Zhipeng Wang. Aligning diffusion language models via unpaired preference optimization. *arXiv preprint arXiv:2510.23658*, 2025. URL https://arxiv.org/abs/2510.23658.

Seungjae Jung, Gunsoo Han, Daniel Wontae Nam, and Kyoung-Woon On. Binary classifier optimization for large language model alignment. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1858–1872, 2025. doi: 10.18653/v1/2025.acl-long.93. URL https://aclanthology.org/2025.acl-long.93/.

Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2305.20081.

Diederik P. Kingma and Ruiqi Gao. Understanding diffusion objectives as the ELBO with simple data augmentation. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/ce79fbf9baef726645bc2337abb0ade2-Abstract-Conference.html.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*, 2022. URL https://iclr.cc/virtual/2022/poster/5941.

Aviral Kumar, Xue Bin Peng, and Sergey Levine. Reward-conditioned policies. *arXiv preprint arXiv:1912.13465*, 2019. URL https://arxiv.org/abs/1912.13465.

Qiyang Li and Sergey Levine. Q-learning with adjoint matching. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2601.14234.

Shufan Li, Konstantinos Kallidromitis, Akash Gokul, Yusuke Kato, and Kazuki Kozuka. Aligning diffusion models by optimizing human utility. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2404.04465.

Zhanhao Liang, Yuhui Yuan, Shuyang Gu, Bohan Chen, Tiankai Hang, Mingxi Cheng, Ji Li, and Liang Zheng. Aesthetic post-training diffusion models from generic preferences with step-by-step preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. URL https://arxiv.org/abs/2406.04314.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.02747.

Huihan Liu, Soroush Nasiriany, Lance Zhang, Zhiyao Bao, and Yuke Zhu. Robot learning on the job: Human-in-the-loop autonomy and learning during deployment. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.005. URL https://www.roboticsproceedings.org/rss19/p005.html.

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*, 2025a. URL https://arxiv.org/abs/2505.05470.

Jie Liu, Gongye Liu, Jiajun Liang, Ziyang Yuan, Xiaokun Liu, Mingwu Zheng, Xiele Wu, Qiulin Wang, Menghan Xia, Xintao Wang, Xiaohong Liu, Fei Yang, Pengfei Wan, Di Zhang, Kun Gai, Yujiu Yang, and Wanli Ouyang. Improving video generation with human feedback. *arXiv preprint arXiv:2501.13918*, 2025b. URL https://arxiv.org/abs/2501.13918.

Zhi Liu. CrossVLA: Cross-paradigm post-training and inference optimization for vision-language-action models. *arXiv preprint arXiv:2605.21854*, 2026. URL https://arxiv.org/abs/2605.21854.

Jianlan Luo, Perry Dong, Yuexiang Zhai, Yi Ma, and Sergey Levine. RLIF: Interactive imitation learning as reinforcement learning. In *International Conference on Learning Representations*, 2024a. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/9c537882044c8b5352c363e840872ddb-Abstract-Conference.html.

Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *arXiv preprint arXiv:2410.21845*, 2024b. URL https://arxiv.org/abs/2410.21845.

Yiming Mao, Zixi Yu, Weixin Mao, Yinhao Li, Qirui Hu, Zihan Lan, Minzhao Zhu, and Hua Chen. ARM: Advantage reward modeling for long-horizon manipulation. *arXiv preprint arXiv:2604.03037*, 2026. URL https://arxiv.org/abs/2604.03037.

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. URL https://arxiv.org/abs/2507.21053.

Marco Moletta, Michael C. Welle, and Danica Kragic. Preference aligned visuomotor diffusion policies for deformable object manipulation. *arXiv preprint arXiv:2602.09583*, 2026. URL https://arxiv.org/abs/2602.09583.

Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. *arXiv preprint arXiv:2410.13816*, 2024. URL https://arxiv.org/abs/2410.13816.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022. URL https://arxiv.org/abs/2203.02155.

Seohong Park, Qiyang Li, and Sergey Levine. Flow Q-learning. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=KVf2SFL1pi.

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. URL https://arxiv.org/abs/1910.00177.

Physical Intelligence et al. $\pi^*_{0.6}$: A VLA that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025. URL https://arxiv.org/abs/2511.14759.

Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via Q-score matching. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2312.11752.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2305.18290.

Allen Z. Ren, Justin Lidard, Lars Lien Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2409.00588.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635. PMLR, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Marta Skreta, Lazar Atanackovic, Avishek Joey Bose, Alexander Tong, and Kirill Neklyudov. The superposition of diffusion models using the Itô density estimator. *arXiv preprint arXiv:2412.17762*, 2025. URL https://arxiv.org/abs/2412.17762.

Andrew Wagenmaker, Yunchu Zhang, Mitsuhiko Nakamoto, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 258–282. PMLR, 2025. URL https://proceedings.mlr.press/v305/wagenmaker25a.html.

Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. URL https://arxiv.org/abs/2311.12908.

Hao Wang, Jiuzhou Lei, Dayou Li, Bangya Liu, Minghui Zheng, Manling Li, Ruohan Zhang, and Zhiwen Fan. Behavior uncloning: Distilling mode redirection into policy weights without inference-time steering. *arXiv preprint arXiv:2606.29201*, 2026a. URL https://arxiv.org/abs/2606.29201.

Yi Wang et al. Learning while deploying: Fleet-scale reinforcement learning for generalist robot policies. *arXiv preprint arXiv:2605.00416*, 2026b. URL https://arxiv.org/abs/2605.00416.

Zhendong Wang, Jonathan J. Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=AHvFDPi-FA.

Yihao Wu, He Zhang, Junbo Tan, Xueqian Wang, and Zhengyou Zhang. FlowPRO: Reward-free reinforced fine-tuning of flow-matching VLAs via proximalized preference optimization. *arXiv preprint arXiv:2606.05468*, 2026. URL https://arxiv.org/abs/2606.05468.

Wenke Xia, Yichu Yang, Hongtao Wu, Xiao Ma, Tao Kong, and Di Hu. Human-assisted robotic policy refinement via action preference optimization. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1235. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/347110fb894281e5e937f6ccd998a6eb-Abstract-Conference.html.

Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, and Ping Luo. DanceGRPO: Unleashing GRPO on visual generation. *arXiv preprint arXiv:2505.07818*, 2025. URL https://arxiv.org/abs/2505.07818.

Brent Yi, Hongsuk Choi, Himanshu Gaurav Singh, Xiaoyu Huang, Takara E. Truong, Carmelo Sferrazza, Yi Ma, Rocky Duan, Pieter Abbeel, Guanya Shi, Karen Liu, and Angjoo Kanazawa. Flow policy gradients for robot control. *arXiv preprint arXiv:2602.02481*, 2026. URL https://arxiv.org/abs/2602.02481.

Shiyuan Zhang, Weitong Zhang, and Quanquan Gu. Energy-weighted flow matching for offline reinforcement learning. In *International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=HA0oLUvuGI.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. ReinFlow: Fine-tuning flow matching policy with online reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025b. doi: 10.52202/085713-3547. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/98d6f928497d9eaad9f320fb2db3040d-Abstract-Conference.html.

Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. DiffusionNFT: Online diffusion reinforcement with forward process. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/d8e68ddfe22520b45b8fb8d5cbde5a21-Abstract-Conference.html.

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. URL https://arxiv.org/abs/2606.11087.

A Experimental Details

A.1 Task Protocol and Feedback Data

Robot and workspace. Experiments use a stationary upper-body bimanual manipulation platform at a 75 cm-high table. It provides 43 actuated DoF through two 7-DoF arms, two 12-DoF dexterous hands, and a 5-DoF neck–waist chain. The upper body is 792 mm high; each arm is 717 mm long, and the 1.66 m arm span (excluding hands) covers a nominal 2.1 m-diameter manipulation workspace. The base, neck, and waist remain fixed throughout evaluation. A table-fixed 60×40 cm active rectangle is centered on the platform. Three synchronized RGB views (head and two wrists) and proprioceptive state form the policy observation. The policy issues bimanual Cartesian and hand commands under receding-horizon execution. Built-in joint limits and an independent emergency stop remain active throughout evaluation.

Table 2: Frozen task protocols. Dimensions are measured in the table frame.

Task	Reset and randomization	Strict success	Task stages
Flower–Vase Insertion	35 ± 3 cm flower (6–10 mm stem) in a 30×20 cm patch, yaw $\pm 90^\circ$; 20-cm-tall vase with 7-cm mouth in a 15×15 cm goal patch; gap ≥ 20 cm	Stem depth ≥ 8 cm; bloom above rim; vase displacement < 3 cm; released and stable for 2 s	Grasp; align/insert; release
Pick-and-Scan	Six items (5–18 cm, 0.05–0.50 kg) in a 45×30 cm cart; balanced barcode faces; scanner volume $20 \times 20 \times 15$ cm and 40×25 cm post-scan tray on balanced sides	Exactly one valid scanner event per SKU between pickup and tray contact; no wrong or duplicate event; all six items in the tray and stable for 2 s	Six pick–present–place cycles
Bagging	The same six items in a staging tray; open 32-cm-high paper bag with a 25×18 cm mouth; item poses, bag side, and yaw $\pm 15^\circ$ balanced	All six items lie below the rim and are supported by the bag; none remains outside or on the rim; bag upright and stable for 3 s	Six pick–insert–release cycles
Popcorn Scooping	30×20 cm source basket with 120 ± 2 g leveled to approximately 7 cm; 250 mL scoop in a 20×10 cm patch, yaw $\pm 90^\circ$; 18-cm-inner-diameter bowl; gap ≥ 20 cm	Bowl gains 6–10 g; spill ≤ 1.0 g; scoop withdrawn; bowl and basket displacement < 3 cm and stable for 2 s	Tool acquisition; load; transport; deposit/ withdraw

For each run, we record the data sources, dataset splits, policy checkpoint, observation and action interfaces, and labeler version.

Each layout is reset from its manifest: object centers are within 1 cm and 5° of their prescribed poses, containers within 0.5 cm and 3°, the robot returns to one neutral keyframe with open hands, scanner logs are cleared, and both 0.1-g-resolution scales are zeroed. The timeout is 45 s for Flower–Vase Insertion, 180 s for Pick-and-Scan, 150 s for Bagging, and 90 s for Popcorn Scooping. Timing begins with the first non-neutral command and stops early after the terminal stability interval. A safety stop, takeover, object leaving the workspace, or irreversible task error is a failure. A verified camera, network, scanner, or scale fault unrelated to the policy is discarded and rerun from the same layout at the end of the evaluation block.

For scanning, two identical reads within 300 ms form one event cluster; reads of the same SKU separated by at most 0.75 s remain one cluster, whereas a later reappearance is a duplicate event. An event is valid only after the item has left its source support by 3 cm and before its first contact with the post-scan tray. For Popcorn Scooping, bowl gain and spill-tray mass provide the terminal readout.

Unary labels. The frozen initial policy generates every feedback episode autonomously. We give Seedance 2.0 Pro the instruction, task rubric, stage and event context, and synchronized head and wrist clips spanning the action chunk. It returns a desirable, undesirable, or invalid label with an error category and supporting evidence; the labeler never proposes a replacement action.

Each record is decoded twice with normal and reversed view order. A disagreement triggers an additional pass and a majority decision. Records with schema errors, missing or unsynchronized frames, insufficient visual evidence, or conflicts with task sensors are marked invalid and excluded rather than relabelled by a human. Training reads the frozen export offline: desirable and undesirable records provide the feedback chunks, while unlabelled BC records supply the anchor role. Feedback windows are mapped to training chunks using the latency-aware start range in Eq. (13); each retained chunk inherits its annotation's unary label.

Table 3: Frozen feedback-collection and evaluation budget. Pos./Neg. are unique training-chunk quotas after held-out anchors and their exclusion neighborhoods are removed; Eval. is the number of trials per policy.

Task	Min. rollouts	Pos. quota	Neg. quota	Eval.
Flower–Vase Insertion	120	420	180	25
Pick-and-Scan	140	500	260	25
Bagging	160	560	320	25
Popcorn Scooping	140	500	280	25

Collection continues beyond the minimum rollout count until both label quotas are met. Chunks are sampled without replacement by source episode; the frozen manifest records additional rollouts and exclusions.

Training configuration. The four task-specific runs use the fixed recipe below. Each run mixes its scene variants and trains one checkpoint from the same pretrained initialization; the trainable policy and frozen reference begin from identical weights.

Flow-KTO minimizes Eq. (17), averaging the bounded sigmoid losses separately over desirable and undesirable chunks before applying λ_+ and λ_-. The feedback score uses the squared CFM error with the truncated time weighting in Eq. (7): $t \sim \mathcal{U}(0, 1-\delta)$ with weight $(1-\delta)t/(1-t)$. Current and frozen-reference losses share the sampled flow times and noise. The drift estimate is the minibatch mean in Eq. (12), refreshed using current-policy action samples and treated as a constant for feedback gradients. The BC anchor uses the standard CFM loss in Eq. (6) on the fixed BC pool.

Table 4: Implementation summary. The same configuration is used across task-specific runs.

Component	Configuration
Policy	Qwen3-VL-2B-Instruct with a flow-matching action expert; all parameters are fine-tuned
Interface	Three RGB views and proprioceptive state; bimanual Cartesian delta actions with receding-horizon execution
Training	Desirable, undesirable, and BC chunks with a frozen reference, drift centering, and a BC anchor; shared optimization settings, with training duration specified separately for ablations
Checkpoint	The final scheduled checkpoint is fixed before robot evaluation; robot outcomes are not used for model selection

SFT minimizes the standard CFM loss in Eq. (6). It uses the same desirable and BC pools and the same optimization schedule as Flow-KTO, replacing undesirable chunks with BC examples. Its final checkpoint is fixed before robot evaluation.

Comparison protocols. FPO is evaluated as a fixed-data variant of its clipped flow-policy objective (McAllister et al., 2025), following the fixed-data comparison setting used for the PPO baseline in $\pi^*_{0.6}$ (Physical Intelligence et al., 2025). Each frozen deployment chunk's desirable or undesirable label $s_i \in \{+1, -1\}$ replaces the advantage in the clipped update. Policy optimization uses no fresh robot rollouts or value-function advantages. Conditioned BC uses the same frozen deployment chunks with their labels as a binary conditioning input and selects the desirable condition at evaluation. Both methods start from Flow-KTO's pretrained initial policy and follow the common evaluation protocol.

A.2 Metrics and Evaluation Protocol

SR and TMR follow the task-attempt definitions in Eq. (18). Strict success applies the complete rubric in Table 2. TMR counts each attempt at most once; attempts that terminate early remain in the denominator.

Throughput measures successful task executions per hour. For each task, we compute

$$\mathrm{Throughput}_q(\pi) = 3600\,\frac{N_q^{\mathrm{success}}}{T_q}, \tag{19}$$

where N_q^{success} is the number of strictly successful trials and T_q is total evaluation execution time in seconds. Our timing protocol includes all successful and failed trials, from the first non-neutral command until success,

failure termination, or timeout, including inference and execution waits. Manual scene reset time and verified external technical faults are excluded. Each success counts one complete task execution; for Pick-and-Scan and Bagging, all six items must satisfy the task rubric. Throughput is reported separately for each task.

Each task uses 10 fixed layouts derived from initial-policy failures and 15 randomized layouts sampled from the valid task workspace. The evaluation manifest is fixed before any policy is tested and shared across policies. It balances discrete task conditions and rejects invalid continuous configurations, such as object overlap or workspace violations. Policy order is counterbalanced, and policies evaluated on the same layout receive identical stochastic conditions.

Checkpoint identity is masked during scoring whenever possible, and evaluation outcomes are not used for checkpoint selection. Binary rates from the 25-trial design are reported with 95% Wilson intervals. The controls without latency alignment or the BC anchor, and the Popcorn Scooping control without drift estimation at epoch 3, report summary rates with fluctuation ranges. Policy comparisons use layout-level paired bootstrap intervals and McNemar's test; macro averages give equal weight to each task.

A.3 Qualitative Execution Comparison

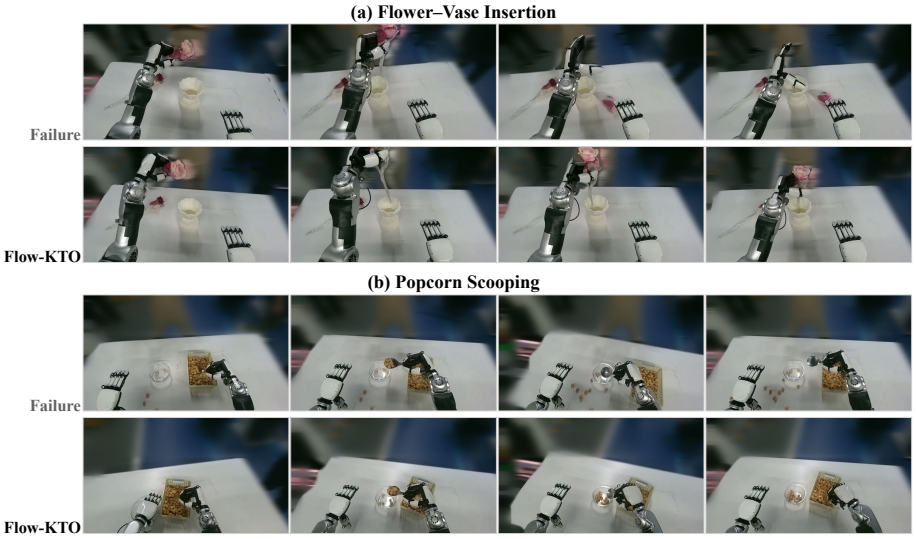

Figure 5: **Learning from recurring failure modes.** Initial-policy failure (top) and Flow-KTO (bottom) at the stem-insertion and scoop-transport decisions. These examples illustrate the complete method's behavior.

A.4 Supplementary Ablation Controls

Figure 4 summarizes the main ablation comparisons. Figures 6, 7, and 8 show both rate and throughput metrics for Flower–Vase Insertion and Popcorn Scooping. Rate whiskers follow the definitions in Appendix A.2. Throughput whiskers show the specified ± 10 successful tasks/h fluctuation bounds; these bounds are not confidence intervals. The objective ablations modify Eq. (17): removing negative feedback sets $\lambda_- = 0$; removing drift centering sets $z = 0$ while retaining the frozen-reference score g_θ; removing the BC anchor sets $\lambda_{\mathrm{BC}} = 0$. The remaining terms retain their class-wise averaging and weights. Training-duration comparisons are specified separately in the main text. The latency control instead sets $k = 0$ in Eq. (13), changing the feedback-chunk construction from the same source annotations while retaining the objective and schedule. In Figure 4, its SR bounds extend 3 percentage points farther below and above the full method's corresponding deviations from the mean; its TMR bounds extend 5.1 points farther on each side. A supplementary norm-matched control removes the negative-feedback term and rescales the remaining update to match the full method's update norm. This control examines whether update magnitude alone accounts for the difference from the full method. Evaluation layouts are excluded from training.

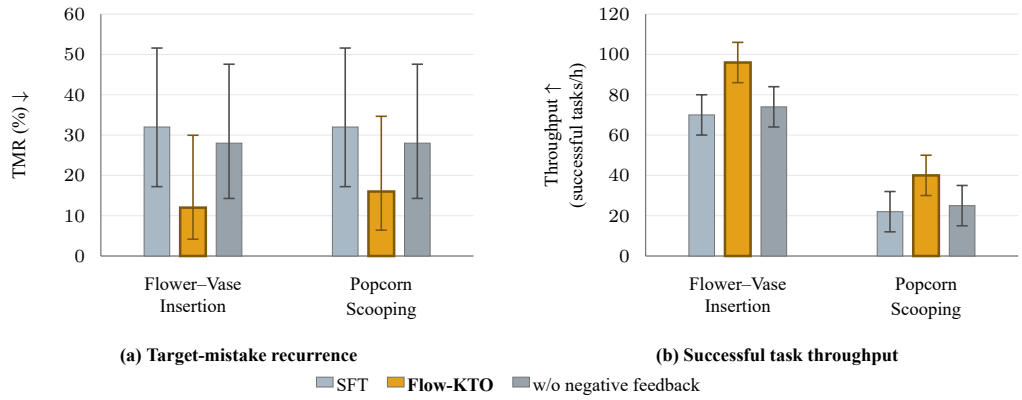

Figure 6: **Negative-feedback ablation.** TMR and throughput on Flower–Vase Insertion and Popcorn Scooping. Negative feedback reduces target-mistake recurrence and increases successful task throughput.

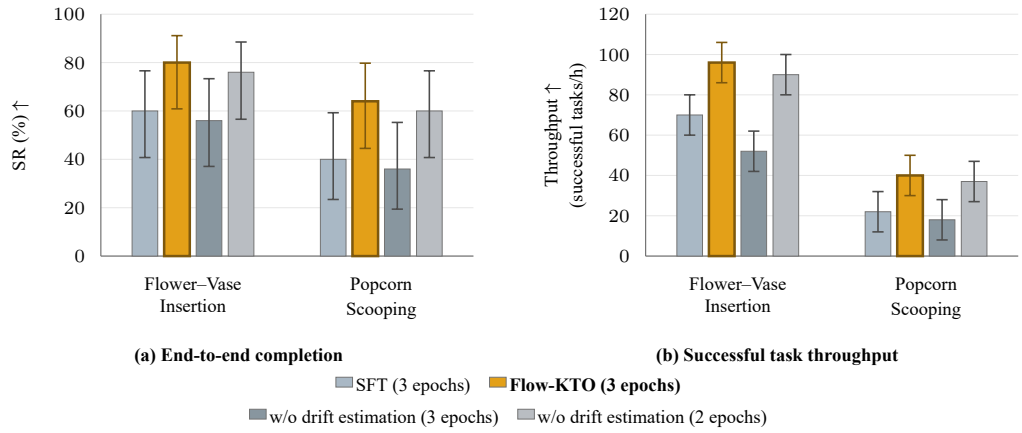

Figure 7: **Drift-estimation ablation.** SR and throughput on Flower–Vase Insertion and Popcorn Scooping. Without drift estimation, both metrics drop sharply from epoch 2 to epoch 3. Full Flow-KTO at epoch 3 outperforms both checkpoints without drift estimation.

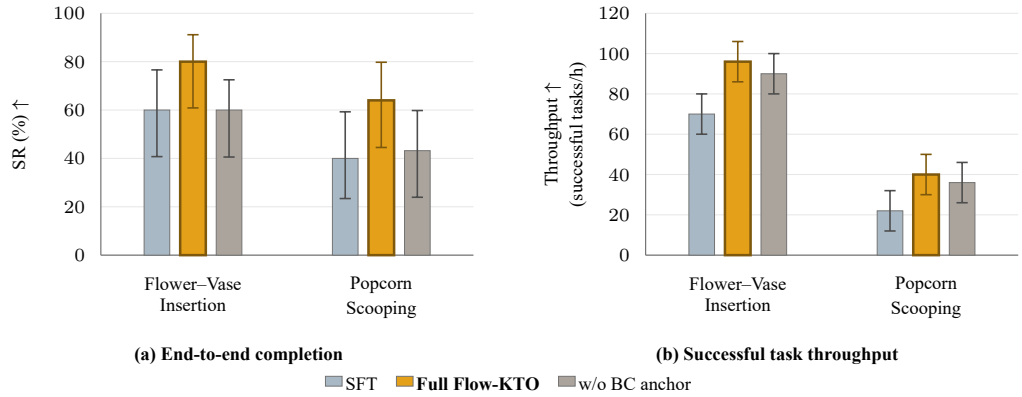

Figure 8: **BC-anchor ablation.** SR and throughput for SFT, full Flow-KTO, and the variant without the BC anchor on Flower–Vase Insertion and Popcorn Scooping. Removing the BC anchor lowers task success and successful task throughput.

A.5 SUPPLEMENTARY EXECUTIONS

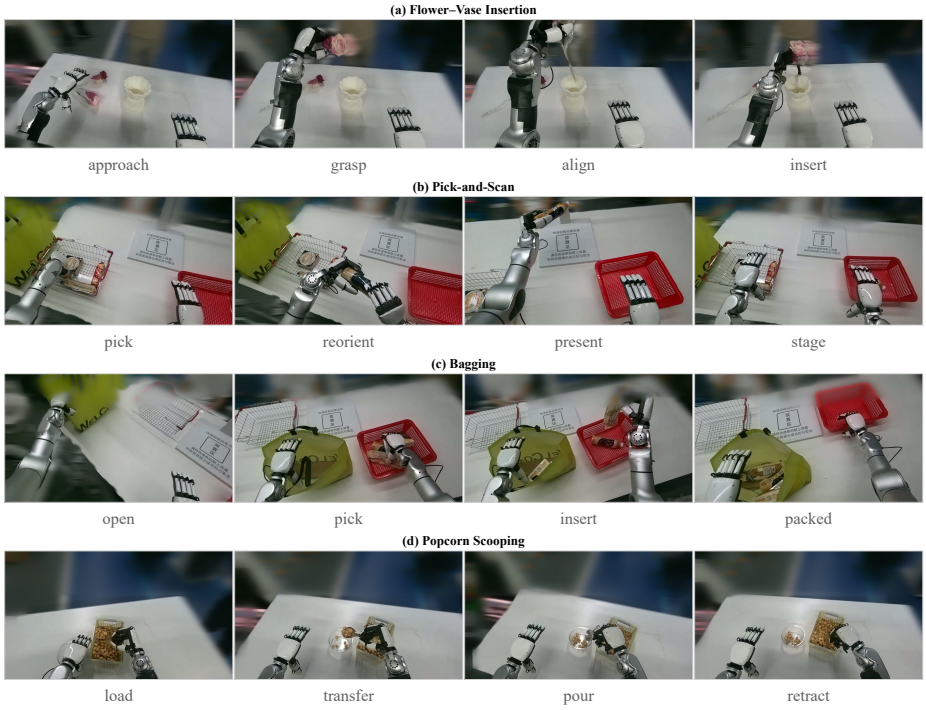

Figure 9: **Representative Flow-KTO executions.** Task-aligned action sequences for all four evaluation tasks.