Q-GUIDED ACTION-TRANSPORT EDITING FOR REAL-WORLD REINFORCEMENT LEARNING OF VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow-based vision-language-action (VLA) policies generate complete action chunks by transporting Gaussian noise through a learned velocity field. Their multi-step generation process provides a strong prior for robot manipulation, but makes online adaptation difficult: task feedback must improve the action without discarding the pretrained transport. We introduce Q-Transport, an off-policy adaptation framework that learns a stochastic editor for partially formed actions at interior flow states. Given an executed action from replay, Q-Transport uses the affine flow path to construct an interior state, applies a noise-conditioned local edit, and maps the edited state to a clean action with the pretrained x-prediction function. The resulting action change is shaped by the flow's state-dependent transport, allowing action-value gradients to reach the editor through the edited flow input. The editor is optimized with a value-and-entropy objective, while successful executed actions can further improve the base policy. This design retains the pretrained action generator while enabling local task-driven adaptation. Experiments on five real-world manipulation tasks show that Q-Transport achieves the best final success rate on every task, with a mean success rate of 83.2%, and reduces human intervention during late online training.

1 INTRODUCTION

Vision-language-action (VLA) models learn generalist robot policies from large and heterogeneous demonstration datasets (Black et al., 2024; Intelligence et al., 2025b). Yet a pretrained policy can remain unreliable on a specific manipulation task (Intelligence et al., 2025a). Online reinforcement learning (RL) uses interaction rewards to improve task performance (Guo et al., 2025; Tan et al., 2025; Lu et al., 2025), but real-robot interaction is costly (Luo et al., 2024b; Lei et al., 2025). The adaptation must therefore make useful task-specific changes from limited experience while retaining the pretrained policy as a source of coherent actions.

This problem is particularly structured for flow-based VLAs. Their action chunks are endpoints of trajectories that transport Gaussian noise through a learned velocity field. Reward is assigned to the executed action, whereas a change to the generator can affect the entire sampling trajectory (Domingo i Enrich et al., 2025; Li & Levine, 2026; Chen et al., 2025a). Directly propagating an action-value gradient through the sampler requires differentiation through its iterative computations (Park et al., 2025). A lightweight alternative is to learn a local edit while keeping the pretrained flow available to construct the action. Where that edit enters the trajectory determines how much of the flow acts on it and how the critic's gradient reaches the editor.

Existing lightweight adaptation methods either modify a completed action or alter the input to the generator (Dong et al., 2026a; Wagenmaker et al., 2025b). This leaves partially formed flow states underused as an editing interface, even though their mapping to the clean action still depends on the pretrained velocity field.

We introduce Q-Transport to edit at this interface. For the affine flow path, an interior state and its predicted velocity define an x-prediction of the clean action. Q-Transport applies a stochastic, noise-level-conditioned edit to the interior state *before* this prediction. The resulting action change depends on the velocity field evaluated at the edited input, so the pretrained model acts as an action-

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.

transport operator rather than merely a source of base actions. A task critic scores the predicted action, and its gradient reaches the editor through this frozen x-prediction map.

During online training, we sample an interior noise level, affinely corrupt an executed action from replay, and train a shared editor using one x-prediction evaluation of its edited input. The edit objective combines action value with adaptive entropy regularization. At inference, each Gaussian seed yields a base action and an edited action: the edited branch successively re-corrupts its current action estimate, applies the editor at interior levels, and updates the estimate by x-prediction. A target critic selects among the completed base and edited candidates. Successful executed actions also provide behavior-cloning updates to the base VLA.

In summary, our contributions are threefold. (1) We introduce Q-Transport, a stochastic editor of interior flow states that produces a clean action proposal with one x-prediction call per training edit. (2) We develop an interior action-transport framework that trains this editor with value guidance and adaptive entropy regularization while retaining the pretrained flow as a differentiable operator between local edits and complete actions. (3) On five real-world manipulation tasks, as summarized in Fig. 1b, Q-Transport achieves the highest final success rate on every task and improves the mean by 13.2 percentage points over the strongest competing method; on four measured tasks, it also reduces human intervention relative to the strongest action-residual baseline in the final quarter of online training.

2 Related Work

2.1 Reinforcement Learning for VLAs

Reinforcement learning for vision-language-action models follows two directions: direct policy post-training and RL-mediated data improvement. Direct methods optimize a pretrained policy from environment feedback using supervised anchors, consistency-based generation, trajectory objectives, sparse rewards, value-calibrated warm-up, or flow-specific likelihoods (Guo et al., 2025; Chen et al., 2025c; Tan et al., 2025; Lu et al., 2025; Li et al., 2026a; Chen et al., 2025a; Liu et al., 2026; Zhang et al., 2026b; Kim et al., 2025). They span autoregressive, diffusion, and flow action distributions, including generalist flow VLAs pretrained at scale (Black et al., 2024; Intelligence et al., 2025b); key choices are the update scope, stabilization mechanism, and policy objective. Data-centric methods generate successful and diverse trajectories for training or distillation (Yang et al., 2025b;a; Xiao et al., 2026), or incorporate rollouts, demonstrations, and interventions through advantage conditioning, implicit imitation, action-level off-policy evaluation, or iterative behavior cloning (Intelligence et al., 2025a; Dong et al., 2025; Yang et al., 2026b). They retain supervised objectives but depend on experience quality, diversity, and coverage.

2.2 Real-World RL for Robotic Manipulation

Real-world robotic RL improves visuomotor policies under sparse rewards, costly failures, resets, and hardware limits. Early systems scaled off-policy RL with autonomous grasping data (Levine et al., 2016; Kalashnikov et al., 2018). Demonstrations, DAgger, and offline replay provide task-specific initialization for online improvement (Ross et al., 2011; Lu et al., 2022; Ball et al., 2023; Hu et al., 2023; Wagenmaker et al., 2025a; Dong et al., 2026c). This reduces unguided exploration but depends on initial coverage and reliable rewards. Human-in-the-loop methods instead retain expert corrections during collection: HG-DAgger labels takeovers, RLiF learns from corrective feedback, and HIL-SERL supports precise manipulation (Kelly et al., 2019; Luo et al., 2024a;b; 2025; Jiang et al., 2025). Residual systems use pretrained controllers with lightweight online correction; Focus-Then-Contact additionally uses affordance rewards and timed interventions for contact-rich tasks (Qiao et al., 2026; Chen et al., 2026). Automated rewards, resets, and distributed collection further reduce physical and human cost (Luo et al., 2024b; Lei et al., 2025). These systems differ in whether supervision only initializes autonomous rollouts or remains active throughout online training.

2.3 Q-Guided Generative Policy Optimization

Q-guided generative policies differ by where value enters sampling. Behavior-constrained methods optimize diffusion policies with Q objectives, reweighting, score regularization, variational im-

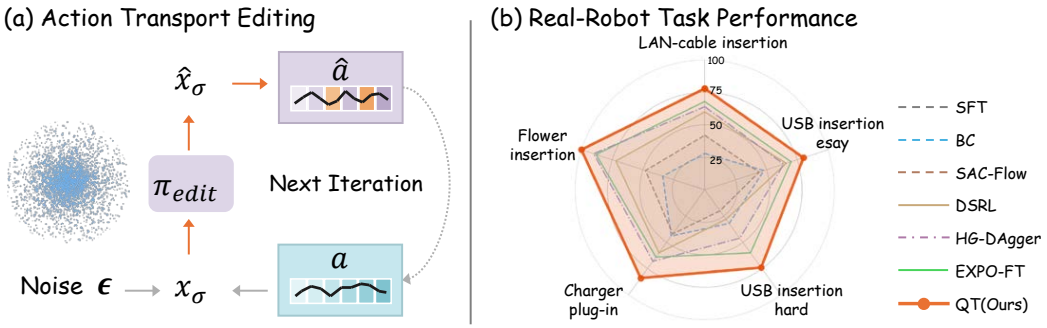


Figure 1: **Q-Transport: action-transport editing and real-robot performance. (a)** At an interior noise level, the editor π_{edit} perturbs the affine-corrupted action state x_σ to obtain $\hat{x}_\sigma$. A single x-prediction evaluation of the frozen flow produces an edited action chunk $\widetilde{a}$, which is re-corrupted for the next iteration. **(b)** Success rates (%) of Q-Transport (QT) and the compared baselines across five real-robot manipulation tasks. Higher is better.

provement, or actor–critic learning (Wang et al., 2022; Chen et al., 2022; Hansen-Estruch et al.; Kang et al., 2023; Chen et al., 2024; Ding et al., 2024; Fang et al., 2025; Wang et al., 2024; Zhang et al., 2024). Output-level methods rank, refine, or distill complete actions (Lu et al., 2023; Mark et al., 2025; Frans et al., 2025; Zhang et al., 2026a). They preserve the sampler at the cost of extra candidates or stages. Denoising-level methods modify scores, velocities, or targets through value guidance, regression, or adjoint gradients (Psenka et al., 2023; Zhang et al., 2025; Domingo i Enrich et al., 2025; Dong et al., 2026e; Li & Levine, 2026; Li et al., 2026b; Wang & Yu, 2025; Li et al., 2026c; Yang et al., 2026a; Zhou et al., 2026). They affect the full trajectory but retrain or differentiate through iterative components. Short-path methods use one-step flows, learned noise distributions, or early rejection (Park et al., 2025; Chen et al., 2025b; Wagenmaker et al., 2025b; Dong et al., 2026d). The insertion point trades policy context against gradient and inference cost.

2.4 Lightweight Refinement of Pretrained Policies

Residual RL retains a base controller and learns lightweight corrections. Classical methods refine hand-designed or demonstration-derived controllers (Silver et al., 2018; Johannink et al., 2019; Alakuijala et al., 2021; Carvalho et al., 2022; Davchev et al., 2022; Dodeja et al., 2025). Modern behavior-cloning systems correct action chunks or individual steps (Ankile et al., 2025b; Yuan et al., 2025; Ankile et al., 2025a; Chen et al., 2026). Specialists also distill failure and recovery trajectories into a generalist (Xiao et al., 2026). For generative policies, EXPO learns an exploration actor, EXPO-FT adds action-chunk residuals, and DSRL optimizes the source-noise distribution of a frozen diffusion policy (Dong et al., 2026b;a; Wagenmaker et al., 2025b). Compact actor–critic interfaces, latent guidance, and critic reranking provide other backbone-preserving refinements (Xu et al., 2026; Zhang et al., 2026c; Nakamoto et al., 2025). These methods differ in whether auxiliary control acts on source noise, latent states, or completed actions.

3 Method

3.1 Problem Setup

Online adaptation task We adapt a pretrained task-conditioned robot policy with online interaction. At decision step t, the policy receives an observation s_t containing the camera images, proprioception, and task prompt, then executes a complete action chunk $a_t \in \mathbb{R}^{T\times D}$. The environment returns the next observation and a binary task reward $r_t \in \{0, 1\}$, where $r_t = 1$ denotes successful completion of the task at that decision. Each executed transition (s_t, a_t, r_t, s_{t+1}) is added to the replay buffer $\mathcal{D}$. The critic and edit policy condition on the observation features used by the task, while the pretrained VLA retains the full task condition. We write $Q(s_t, a_t)$ for the critic on the learned action subspace; local flow constructions below omit the decision index.

Base policy We use $\pi_{0.5}$ (Intelligence et al., 2025b) as the pretrained base policy VLA. For a normalized action a and $\epsilon \sim \mathcal{N}(0, I)$, it uses the affine probability path (Lipman et al., 2023)

$$x_\sigma = (1-\sigma)a + \sigma\epsilon, \qquad \sigma \in [0,1]. \tag{1}$$

Under this conditional path, the target velocity is $\epsilon - a$; the model $v_\psi(s, x_\sigma, \sigma)$ predicts the corresponding velocity from the intermediate state. Generation starts from Gaussian noise at $\sigma = 1$. On the grid $1 = \sigma_N > \cdots > \sigma_0 = 0$, one Euler step toward the action endpoint is

$$x_{k-1} = x_k + (\sigma_{k-1} - \sigma_k) v_\psi(s, x_k, \sigma_k). \tag{2}$$

3.2 Q-Guided Action-Transport Editing

Action transport as the editing interface The base policy defines more than a distribution over completed actions. It also defines how an intermediate state is converted into an action through its state-dependent velocity field. This makes the location of an intervention a structural choice. A residual added after generation no longer passes through this conversion, whereas a perturbation to the initial noise is transformed by the entire sampler. We instead edit an intermediate state (Fig. 1a), so the learned transport remains part of the action construction while the editor changes the state that the transport receives.

The affine path gives this interface a closed-form endpoint interpretation. For $x_\sigma = (1-\sigma)a + \sigma\epsilon$, the conditional velocity target is $\epsilon - a$, and therefore $x_\sigma - \sigma(\epsilon - a) = a$. We use the corresponding x-prediction map

$$\widehat{a}_{\bar{\psi}}(s; y, \sigma) = y - \sigma v_{\bar{\psi}}(s, y, \sigma). \tag{3}$$

Under the affine flow-matching path, this map predicts the clean action endpoint from the interior state.

From flow states to action edits This interpretation also explains why the edit is applied before x-prediction. For a small state perturbation Δx, the induced change in the predicted action is locally

$$\widehat{a}_{\bar{\psi}}(s; x_\sigma + \Delta x, \sigma) - \widehat{a}_{\bar{\psi}}(s; x_\sigma, \sigma) \approx \left[I - \sigma \nabla_x v_{\bar{\psi}}(s, x_\sigma, \sigma)\right] \Delta x. \tag{4}$$

Thus, the editor residual is transformed according to the local geometry of the base policy and the intervention level.

Given a replay action a and $\sigma \in (0,1)$, the editor samples a local perturbation of its affine-corrupted state:

$$\begin{aligned} x_\sigma &= (1-\sigma)a + \sigma\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I), \\ \Delta x &\sim \pi_{\text{edit},\theta}(\cdot \mid s, x_\sigma, \sigma), \\ \widetilde{a} &= \widehat{a}_{\bar{\psi}}(s; x_\sigma + \Delta x, \sigma). \end{aligned} \tag{5}$$

The resulting edited action is evaluated by the task critic $Q(s, \widetilde{a})$. The editor uses a soft actor–critic objective (Haarnoja et al., 2018):

$$\mathcal{L}_{\text{edit}}(\theta) = \mathbb{E}\left[-Q(s, \widetilde{a}) + \alpha \log \pi_{\text{edit},\theta}(\Delta x \mid s, x_\sigma, \sigma)\right], \tag{6}$$

where the expectation covers replay states, interior noise levels, affine corruption, and stochastic editor actions. The value term selects transported state perturbations according to their estimated action value, while the entropy term preserves a stochastic edit distribution. This objective acts at one noise level; an online policy must compose such edits into complete actions and learn their values from interaction.

3.3 Online Policy Optimization

Composing local edits into a policy At inference, the base policy generates an action with the Euler sampler, while the edited policy repeatedly reuses the affine corruption from Eq. 5, samples a local edit, and applies one x-prediction. With C independent samples of each policy, this produces $2C$ candidates. The conservative target critic selects the highest-valued finite action, $a^\star = \arg\max_{a \in \cup_{i=1}^{C} \{a_i, \widetilde{a}_i\}} Q_{\bar{\phi}}(s, a)$, following critic-based candidate selection in EXPO and EXPO-FT (Dong et al., 2026b;a).

Algorithm 1 Q-Transport sampling and online updates

Require: Replay $\mathcal{D}$; flow v_ψ and target $v_{\bar{\psi}}$; editor $\pi_{\text{edit},\theta}$; critic ensemble Q_ϕ and target $Q_{\bar{\phi}}$; temperature α

function SAMPLING(s)
  Sample $z \sim \mathcal{N}(0, I)$ and set $a \leftarrow z$
  for $k = N, \dots, 1$ **do**
    $a \leftarrow a + (\sigma_{k-1} - \sigma_k) v_{\bar{\psi}}(s, a, \sigma_k)$
  end for
  Sample $z \sim \mathcal{N}(0, I)$ and set $\widetilde{a} \leftarrow z$
  for $k = N-1, \dots, 1$ **do**
    $\sigma \leftarrow k/N$; $x_\sigma \leftarrow (1-\sigma)\widetilde{a} + \sigma\epsilon$
    Sample $\Delta x \sim \pi_{\text{edit},\theta}(\cdot \mid s, x_\sigma, \sigma)$
    $\widetilde{a} \leftarrow x_\sigma + \Delta x - \sigma v_{\bar{\psi}}(s, x_\sigma + \Delta x, \sigma)$
  end for
  return $(a, \widetilde{a})$
end function
while online training **do**
  Sample C paired flow and edited actions with SAMPLING(s)
  Execute the highest-valued finite action under $Q_{\bar{\phi}}$
  Collect an episode and append its executed transitions to $\mathcal{D}$
  for each critic update **do**
    Sample a replay batch from $\mathcal{D}$
    Update ϕ to minimize $\mathbb{E}_{\mathcal{D}}\left[(Q_\phi(s,a) - y_Q)^2\right]$
    $\bar{\phi} \leftarrow (1-\tau_Q)\bar{\phi} + \tau_Q \phi$
  end for
  Update ψ on successful replay actions with $\mathcal{L}_{\text{FM}}$, update $v_{\bar{\psi}}$, and update θ with $\mathcal{L}_{\text{edit}}$
end while

Learning action values Candidate selection requires values for complete actions, but rewards are observed only for actions actually executed. The critic therefore regresses on executed chunks (s_t, a_t) in $\mathcal{D}$, while the same sampler generates $2C$ bootstrap candidates at s_{t+1}. Let $\overline{Q}$ denote the conservative target-critic value, and let $a^\star_{t+1}$ be the highest-valued finite candidate at s_{t+1}. With r_t denoting the observed reward, the TD target and critic loss are

$$\begin{aligned} y_Q &= r_t + \gamma\, \overline{Q}(s_{t+1}, a^\star_{t+1}), \\ \mathcal{L}_Q(\phi) &= \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\left[\left(Q_\phi(s_t,a_t) - y_Q\right)^2\right]. \end{aligned} \tag{7}$$

Optimizing the edit policy The learned values guide the edit at each noise level. For an executed $(s, a) \sim \mathcal{D}$, we sample k uniformly from $\{1, \dots, N-1\}$ and stop gradients through the affine corruption of a. We then sample one edit and evaluate the completed edited action from Eq. 5 under the minimum of the target critic ensemble. In Eq. 6, the value gradient passes through the edited input of the frozen flow to the editor. The entropy term maintains stochastic edits, and α is updated toward a target entropy using the same sample. Each update trains the local operator at one sampled noise level.

Updating the base policy The base policy provides the endpoint prediction used by every edit. Following EXPO, it receives a behavior-cloning update on successful executed actions with the flow-matching objective (Lipman et al., 2023):

$$\mathcal{L}_{\text{FM}}(\psi) = \mathbb{E}_{(s,a)\sim\mathcal{D},\sigma,\epsilon}\left[\left\|v_\psi(s, x_\sigma, \sigma) - (\epsilon - a)\right\|_2^2\right]. \tag{8}$$

Algorithm 1 summarizes the resulting sampling and online updates.

4 EXPERIMENTS

4.1 EXPERIMENTAL DETAILS

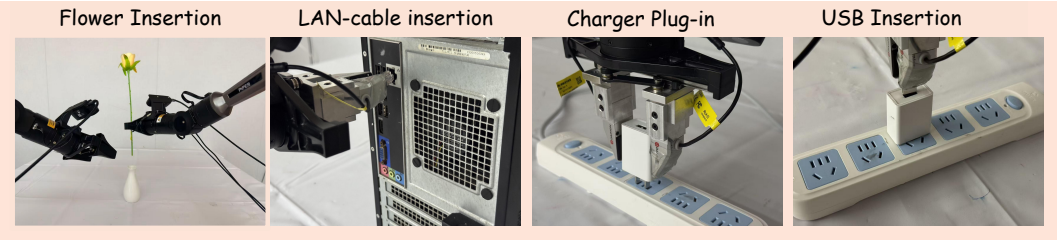


Figure 2: Evaluation tasks with small variations in initial pose. USB insertion includes easy and hard variants.

Tasks As shown in Figure 2, we evaluate five real-world manipulation tasks: flower insertion, LAN-cable insertion, charger plug-in, and easy and hard variants of USB insertion. These tasks involve object handling and contact-rich insertion with small variations in initial pose around each task's nominal setup. The robot setup, task-specific initial-pose perturbations, success criteria, reward definitions, and reset protocols are described in Appendix B.

Baselines We compare Q-Transport against four online adaptation baselines: (1) HG-DAgger, an interactive imitation-learning method that updates the policy from corrections provided by a human expert (Kelly et al., 2019); (2) EXPO-FT, which learns residuals over completed action chunks and uses a critic to select between base and edited proposals (Dong et al., 2026a); (3) DSRL, which learns a policy over the initial-noise space while keeping the base generative policy frozen (Wagenmaker et al., 2025b); and (4) SAC-flow, which performs sample-efficient reinforcement learning of flow-based policies through velocity-reparameterized sequential modeling (Zhang et al., 2026d). SFT and online BC are reported as reference checkpoints. For each task, all methods first start from an offline SFT policy trained on human-collected demonstrations (Intelligence et al., 2025b); the SFT dataset contains 25 demonstration episodes for each task. We then apply online supervised learning with human-in-the-loop (HIL) data, updating only the segments in which the human intervenes, until the task success reaches roughly 30–50%. The resulting task-specific online-BC checkpoint is used to initialize HG-DAgger, Q-Transport, EXPO-FT, DSRL, and SAC-flow. The SFT and BC data-collection durations are summarized in Table 4 and described in Appendix B.4.

Implementation details For the $\pi_{0.5}$-based methods, training uses eight NVIDIA A100 GPUs with 80 GB of memory each, and rollout collection uses one separate NVIDIA A100 80 GB GPU. Q-Transport uses a batch size of 64 for both behavior cloning and online RL. Its critic and edit policy are optimized with AdamW at a learning rate of 3×10^{-4}. All $\pi_{0.5}$-based methods execute action chunks of length $T = 10$ with four Euler steps. SAC-flow uses its own actor–critic architecture and control interface. With eight Gaussian seeds, Q-Transport constructs eight base and eight final edited proposals. Each edited proposal applies three corrections sequentially at $\sigma = 0.75, 0.5, 0.25$, using fresh affine corruption and one x-prediction call per correction. Both action selection and TD backups use these 16 candidates. EXPO-FT also constructs eight base and eight edited proposals at each decision. Appendix A.3 provides the complete network, optimization, and method-specific settings.

4.2 REAL-WORLD RESULTS

Table 1 summarizes the final autonomous success rate on 50 trials for each of the five tasks. Q-Transport obtains the best result on every task, giving a mean success rate of 83.2%. The strongest non-QT method is EXPO-FT, whose mean is 70.0%; thus, Q-Transport improves the average final success rate by 13.2 percentage points. The improvement is also consistent at the task level, with margins of 10, 10, 14, 16, and 10 points over the best competing method on LAN-cable insertion, USB insertion (easy), USB insertion (hard), charger plug-in, and flower insertion, respectively.

Table 1: Final task success rates over 50 autonomous trials per task.

Method	LAN-cable insertion	USB insertion (easy)	USB insertion (hard)	Charger plug-in	Flower insertion	Average
			Success Rate (%)↑			
SFT	0 (0/50)	0 (0/50)	0 (0/50)	0 (0/50)	10 (5/50)	2.0
BC	28 (14/50)	48 (24/50)	32 (16/50)	44 (22/50)	34 (17/50)	37.2
EXPO-FT	68 (34/50)	70 (35/50)	60 (30/50)	64 (32/50)	88 (44/50)	70.0
DSRL	60 (30/50)	66 (33/50)	28 (14/50)	60 (30/50)	72 (36/50)	57.2
HG-Dagger	64 (32/50)	64 (32/50)	46 (23/50)	68 (34/50)	90 (45/50)	66.4
SAC-flow	42 (21/50)	44 (22/50)	20 (10/50)	42 (21/50)	48 (24/50)	39.2
QT	**78 (39/50)**	**80 (40/50)**	**74 (37/50)**	**84 (42/50)**	**100 (50/50)**	**83.2**

The results cover both relatively regular and contact-sensitive insertion settings. On the hard USB task, Q-Transport reaches 74%, compared with 60% for EXPO-FT and 28% for DSRL. On flower insertion, it reaches 100%, while the strongest competing result is 90%. The same ordering is preserved on the remaining three tasks, indicating that the advantage is not due to a single particularly favorable task. Relative to the online-BC reference, Q-Transport raises the mean success rate from 37.2% to 83.2%, showing the benefit of the proposed transport-and-editing procedure after the common BC initialization.

We focus on two questions on USB insertion (hard) and LAN-cable insertion: whether learned edits improve action generation, and whether the three-level correction schedule improves over a single correction. We compare the full method with two variants, each trained independently from the same task-specific online-BC checkpoint. All three systems retain 16 candidates, the same interaction budget, critic-update schedule, and successful-action base-policy updates. Candidate-generation changes apply to rollout collection, TD backups, and final evaluation. All variants follow the evaluation and reporting protocol in Section 4.2.

Without learned edits We set $\Delta x = 0$ at all three levels and omit editor and entropy-temperature updates. The variant retains the same fresh affine corruption and three x-prediction calls at $\sigma = 0.75, 0.5, 0.25$. Each initial seed produces one base action and one final reprocessed action, giving eight of each for critic selection. This comparison measures the contribution of learned edits beyond the same corruption-and-prediction procedure, while holding candidate count and velocity-evaluation count fixed.

Single correction We replace the three-level schedule with one correction at $\sigma = 0.5$, fixed before evaluation. The editor is trained only at this level and uses the same architecture and optimizer-update count as the full method. Both systems retain eight base and eight edited candidates. This comparison tests whether the full schedule's additional predictions yield a useful performance gain over a simpler one-level alternative. It evaluates the schedule as a whole, rather than separating correction count from noise-level coverage.

We report final success rates and learning curves against low-level control steps, together with decision latency for the full and single-correction systems.

4.3 Human-Intervention Efficiency

To examine how much human supervision is needed during online adaptation, we track the human intervention rate, defined as the fraction of control frames in which the operator must intervene. Lower intervention rates indicate that the policy can execute a larger fraction of the rollout autonomously. Figure 3 compares Q-Transport with EXPO-FT (labeled "EXPO" in the plot) on four tasks; the offline SFT dataset and the online-BC initialization are the same as in the main comparison.

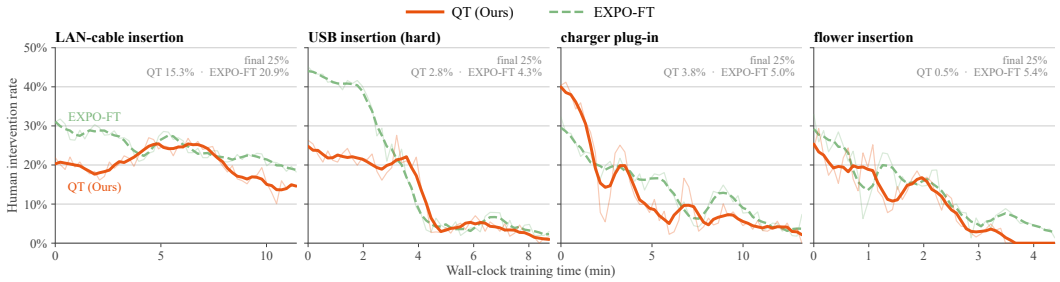
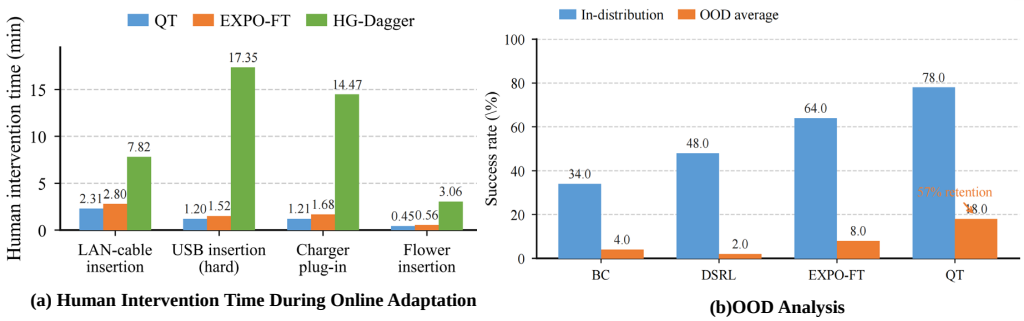


Figure 3: Human intervention rate during online training for Q-Transport (QT) and EXPO-FT (shown as EXPO). The annotations report the mean intervention rate over the final 25% of training. Lower values indicate less human intervention.

In the final 25% of training, Q-Transport has a lower intervention rate on every plotted task: 15.3% versus 20.9% on LAN-cable insertion, 2.8% versus 4.3% on hard USB insertion, 3.8% versus 5.0% on charger plug-in, and 0.5% versus 5.4% on flower insertion. Averaged over these four tasks, the intervention rate is 5.6% for Q-Transport and 8.9% for EXPO, a reduction of 3.3 percentage points (about 37% relative). Under the shared SFT and BC initialization, this indicates less frequent human correction during the late stage of online training. Because intervention rate is a fraction of control frames, these curves alone do not measure the total volume of human-labeled data collected across training.



(a) Human Intervention Time During Online Adaptation

(b)OOD Analysis

Figure 4: Human-intervention time during online training. Values are shown in minutes; lower values indicate less human supervision.

The time comparison in Figure 4 shows the larger operator burden of HG-Dagger under the same task protocol. Its human intervention time is 7.82, 17.35, 14.47, and 3.06 minutes on LAN-cable insertion, hard USB insertion, charger plug-in, and flower insertion, respectively. These values are higher than both QT and EXPO-FT on every task, which is consistent with the expert-correction loop used by this interactive imitation-learning baseline (Kelly et al., 2019).

The comparison between QT and EXPO-FT is more modest but has the same direction on all four tasks: QT uses 2.31 vs. 2.80 minutes on LAN-cable insertion, 1.20 vs. 1.52 minutes on hard USB insertion, 1.21 vs. 1.68 minutes on charger plug-in, and 0.45 vs. 0.56 minutes on flower insertion. Averaged over the four tasks, QT requires 1.29 minutes of human intervention compared with 1.64 minutes for EXPO-FT, a 21.1% reduction. Thus, the figure supports a consistent but moderate reduction in human supervision relative to EXPO-FT, while the much larger gap to HG-Dagger highlights the practical cost of methods that rely on frequent expert corrections.

4.4 Ablations and Out-of-Distribution Generalization

We ablate on LAN-cable insertion, USB insertion (hard), and charger insertion, asking what base-policy consolidation contributes on top of critic-guided editing, how many Euler steps N deployment needs, and how the method degrades from unseen initial positions. Each variant trains from the same

Table 2: Ablations. Success rate (%) over 50 trials per task and configuration. The upper block ablates base-policy consolidation at the training budget $N = 4$; the lower block sweeps the deployment-time Euler steps with all policies trained at $N = 4$. Differences below the ± 11.5-point per-cell interval are not significant.

Configuration	LAN	USB-hard	Charger	Avg.
BC (base VLA only)	28	32	44	34.7
Component ablation				
w/o base-policy consolidation	66	62	74	67.3
Q-Transport (full)	**78**	**74**	**84**	**78.7**
Deployment Euler steps (all trained at $N = 4$)				
$N = 2$	56	50	64	56.7
$N = 4$ (default)	78	74	84	78.7
$N = 8$	80	76	84	80.0
$N = 10$	80	76	86	80.7

task-specific online-BC checkpoint under the settings of Table 3; the N sweep changes deployment only. The row *BC (base VLA only)* is the shared initialization. At 50 trials the 95% interval on one cell is ± 11.5 points, so the component claim rests on the pooled comparison ($n = 150$).

Base-policy consolidation Disabling the update that regresses the base policy toward the critic-selected actions of successful episodes costs 11.4 points on average (-12, -12, -10). No per-task gap clears the ± 11.5-point interval, but the pooled gap exceeds its 95% interval of ± 10.0 points. Since the editor acts on the base policy's own trajectory, freezing the base leaves it correcting a static, mis-centered proposal distribution. The ablated variant still reaches 67.3, so critic-guided editing alone accounts for most of the gain over the 34.7 initialization.

Deployment-time Euler steps The knee sits at the training budget: $N = 2 \to 4$ buys 22.0 points (11.0 per added evaluation) while $N = 4 \to 8$ buys 1.3 (0.33), a $33\times$ difference in marginal return. Only the former clears the noise floor, so we do not claim the $+1.3$ and $+2.0$ points at $N = 8$ and $N = 10$. The $N = 2$ collapse is structural: retaining only the interior node $\sigma = 0.5$ – whose mean critic improvement is negative ($\overline{\Delta Q} = -1.4 \times 10^{-3}$) – discards the two productive correction levels. We adopt $N = 4$ by default.

Out-of-distribution initial positions On four held-out initial positions per task, 20 trials each ($n = 240$ pooled, Figure 4), all systems degrade sharply from unseen poses, but Q-Transport reaches 18% against 8%, clearing the pooled 95% interval (± 6.0 points), and retains 23% of its in-distribution success against 4–13% for the baselines. Holding on retention, not only absolute success, suggests the advantage is not inherited from the higher starting point. Editing at interior nodes $\sigma \in [0.25, 0.75]$ plausibly sees inputs that shift less under an unseen pose, though we offer this as an interpretation, not an isolated cause.

5 Conclusion

We introduced Q-Transport for online reinforcement learning of flow-based VLA policies. It edits an affine-corrupted executed action at an interior flow state, maps the edit to a clean action with the frozen x-prediction function, and uses a critic to select among base and edited proposals. On five real-world manipulation tasks, Q-Transport achieved the highest final success rate among the evaluated methods on every task, with a mean of 83.2%; ablation results also support base-policy consolidation at the tested training budget. Performance nevertheless degraded on unseen initial positions, indicating that robust generalization remains an open issue.

References

Minttu Alakuijala, Gabriel Dulac-Arnold, Julien Mairal, Jean Ponce, and Cordelia Schmid. Residual reinforcement learning from demonstrations. *arXiv preprint arXiv:2106.08050*, 2021.

Lars Ankile, Zhenyu Jiang, Rocky Duan, Guanya Shi, Pieter Abbeel, and Anusha Nagabandi. Residual off-policy rl for finetuning behavior cloning policies. *arXiv preprint arXiv:2509.19301*, 2025a.

Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement-residual rl for precise assembly. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 01–08. IEEE, 2025b.

Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pp. 1577–1594. PMLR, 2023.

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.

Joao Carvalho, Dorothea Koert, Marek Daniv, and Jan Peters. Residual robot learning for object-centric probabilistic movement primitives. *arXiv preprint arXiv:2203.03918*, 2022.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. *arXiv preprint arXiv:2209.14548*, 2022.

Huayu Chen, Cheng Lu, Zhengyi Wang, Hang Su, and Jun Zhu. Score regularized policy optimization through diffusion behavior. In *International Conference on Learning Representations*, volume 2024, pp. 10211–10230, 2024.

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

Tianyi Chen, Haitong Ma, Na Li, Kai Wang, and Bo Dai. One-step flow policy mirror descent. *arXiv preprint arXiv:2507.23675*, 2025b.

Yuhui Chen, Shuai Tian, Shugao Liu, Yingting Zhou, Haoran Li, and Dongbin Zhao. Conrft: A reinforced fine-tuning method for vla models via consistency policy. *arXiv preprint arXiv:2502.05450*, 2025c.

Zhongxi Chen, Yifan Han, Yanming Shao, Huanming Liu, Congsheng Xu, Xiaoyu Chen, Yao Mu, and Wenzhao Lian. Bora: Bridging offline reinforcement learning and online residual adaptation for real-world dexterous vla models. *arXiv preprint arXiv:2605.30226*, 2026.

Todor Davchev, Kevin Sebastian Luck, Michael Burke, Franziska Meier, Stefan Schaal, and Subramanian Ramamoorthy. Residual learning from demonstration: Adapting dmps for contact-rich manipulation. *IEEE Robotics and Automation Letters*, 7(2):4488–4495, 2022.

Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. *Advances in Neural Information Processing Systems*, 37:53945–53968, 2024.

Lakshita Dodeja, Karl Schmeckpeper, Shivam Vats, Thomas Weng, Mingxi Jia, George Konidaris, and Stefanie Tellex. Accelerating residual reinforcement learning with uncertainty estimation. *IEEE Robotics and Automation Letters*, 11(1):970–977, 2025.

Carles Domingo i Enrich, Michal Drozdzal, Brian Karrer, and Ricky TQ Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. In *International Conference on Learning Representations*, volume 2025, pp. 53791–53846, 2025.

Perry Dong, Alec M Lessing, Annie S Chen, and Chelsea Finn. Reinforcement learning via implicit imitation guidance. *arXiv preprint arXiv:2506.07505*, 2025.

Perry Dong, Kuo-Han Hung, Tian Gao, Dorsa Sadigh, and Chelsea Finn. Expo-ft: Sample-efficient reinforcement learning finetuning for vision-language-action models, 2026a. URL `https://arxiv.org/abs/2605.25477`.

Perry Dong, Qiyang Li, Dorsa Sadigh, and Chelsea Finn. Expo: Stable reinforcement learning with expressive policies. In *International Conference on Learning Representations*, volume 2026, pp. 131379–131394, 2026b.

Perry Dong, Suvir Mirchandani, Dorsa Sadigh, and Chelsea Finn. What matters for batch online reinforcement learning in robotics? In *International Conference on Learning Representations*, volume 2026, pp. 112294–112308, 2026c.

Perry Dong, Alexander Swerdlow, Dorsa Sadigh, and Chelsea Finn. Faster: Value-guided sampling for fast rl. *arXiv preprint arXiv:2604.19730*, 2026d.

Perry Dong, Chongyi Zheng, Chelsea Finn, Dorsa Sadigh, and Benjamin Eysenbach. Value flows. In *International Conference on Learning Representations*, volume 2026, pp. 128614–128646, 2026e.

Linjiajie Fang, Ruoxue Liu, Jing Zhang, Wenjia Wang, and Bingyi Jing. Diffusion actor-critic: Formulating constrained policy iteration as diffusion noise regression for offline reinforcement learning. In *International Conference on Learning Representations*, volume 2025, pp. 75743–75763, 2025.

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

Yanjiang Guo, Jianke Zhang, Xiaoyu Chen, Xiang Ji, Yen-Jen Wang, Yucheng Hu, and Jianyu Chen. Improving vision-language-action model with online reinforcement learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15665–15672. IEEE, 2025.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

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

Hengyuan Hu, Suvir Mirchandani, and Dorsa Sadigh. Imitation bootstrapped reinforcement learning. *arXiv preprint arXiv:2311.02198*, 2023.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. $\pi^*_{0.6}$: a vla that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025a.

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

Yunfan Jiang, Chen Wang, Ruohan Zhang, Jiajun Wu, and Li Fei-Fei. Transic: Sim-to-real policy transfer by learning from online correction. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 1691–1729. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/jiang25a.html.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 international conference on robotics and automation (ICRA)*, pp. 6023–6029. IEEE, 2019.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on robot learning*, pp. 651–673. PMLR, 2018.

Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:67195–67212, 2023.

Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8077–8083. IEEE, 2019.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

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

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

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, 2026a.

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

Qiyang Li, Seohong Park, and Sergey Levine. Decoupled q-chunking. In *International Conference on Learning Representations*, volume 2026, pp. 145315–145389, 2026b.

Zeyang Li, Sunbochen Tang, and Navid Azizan. Reverse flow matching: A unified framework for online reinforcement learning with diffusion and flow policies. *arXiv preprint arXiv:2601.08136*, 2026c.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Jijia Liu, Feng Gao, Bingwen Wei, Xinlei Chen, Qingmin Liao, Yi Wu, Chao Yu, and Yu Wang. What can rl bring to vla generalization? an empirical study. *Advances in Neural Information Processing Systems*, 38:97121–97151, 2026.

Cheng Lu, Huayu Chen, Jianfei Chen, Hang Su, Chongxuan Li, and Jun Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *International Conference on Machine Learning*, pp. 22825–22855. PMLR, 2023.

Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. Vla-rl: Towards masterful and general robotic manipulation with scalable reinforcement learning. *arXiv preprint arXiv:2505.18719*, 2025.

Yao Lu, Karol Hausman, Yevgen Chebotar, Mengyuan Yan, Eric Jang, Alexander Herzog, Ted Xiao, Alex Irpan, Mohi Khansari, Dmitry Kalashnikov, et al. Aw-opt: Learning robotic skills with imitation andreinforcement at scale. In *Conference on Robot Learning*, pp. 1078–1088. PMLR, 2022.

Jianlan Luo, Perry Dong, Yuexiang Zhai, Yi Ma, and Sergey Levine. Rlif: Interactive imitation learning as reinforcement learning. In *International Conference on Learning Representations*, volume 2024, pp. 36329–36351, 2024a.

Jianlan Luo, Zheyuan Hu, Charles Xu, You Liang Tan, Jacob Berg, Archit Sharma, Stefan Schaal, Chelsea Finn, Abhishek Gupta, and Sergey Levine. Serl: A software suite for sample-efficient robotic reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16961–16969. IEEE, 2024b.

Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *Science Robotics*, 10(105):eads5033, 2025.

Max Sobol Mark, Tian Gao, Georgia Gabriela Sampaio, Mohan Kumar Srirama, Archit Sharma, Chelsea Finn, and Aviral Kumar. Policy-agnostic RL: Offline RL and online RL fine-tuning of any class and backbone. In *Scaling Self-Improving Foundation Models without Human Supervision*, 2025. URL https://openreview.net/forum?id=Mevh2AmlPc.

Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 4996–5013. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/nakamoto25a.html.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48104–48127. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/park25f.html.

Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. *arXiv preprint arXiv:2312.11752*, 2023.

Guanren Qiao, Ruixiang Ouyang, Sheng Xu, Ruixing Jin, Yueci Deng, Yunxin Tai, Kui Jia, and Guiliang Liu. Focus-then-contact: Speeding up robotic contact-rich task learning with affordance-guided real-world residual reinforcement learning. In *Forty-third International Conference on Machine Learning*, 2026.

Stéphane 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*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

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

Andrew Wagenmaker, Perry Dong, Raymond Tsao, Chelsea Finn, and Sergey Levine. Posterior behavioral cloning: Pretraining bc policies for efficient rl finetuning. *arXiv preprint arXiv:2512.16911*, 2025a.

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 Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 258–282. PMLR, 27–30 Sep 2025b. URL https://proceedings.mlr.press/v305/wagenmaker25a.html.

Feng Wang and Zihao Yu. Coefficients-preserving sampling for reinforcement learning with flow matching. *arXiv preprint arXiv:2509.05952*, 2025.

Yinuo Wang, Likun Wang, Yuxuan Jiang, Wenjun Zou, Tong Liu, Xujie Song, Wenxuan Wang, Liming Xiao, Jiang Wu, Jingliang Duan, et al. Diffusion actor-critic with entropy regulator. *Advances in Neural Information Processing Systems*, 37:54183–54204, 2024.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Zhengyi Luo, Yuqi Xie, Fengyuan Hu, Jim Fan, Guanya Shi, et al. Self-improving vision-language-action models with data generation via residual rl. In *International Conference on Learning Representations*, volume 2026, pp. 13209–13236, 2026.

Charles Xu, Jost Tobias Springenberg, Michael Equi, Ali Amin, Adnan Esmail, Sergey Levine, and Liyiming Ke. Rl token: Bootstrapping online rl with vision-language-action models. *arXiv preprint arXiv:2604.23073*, 2026.

Rushuai Yang, Zhiyuan Feng, Tianxiang Zhang, Kaixin Wang, Chuheng Zhang, Li Zhao, Xiu Su, Yi Chen, and Jiang Bian. Discover, learn, and reinforce: Scaling vision-language-action pretraining with diverse rl-generated trajectories. *arXiv preprint arXiv:2511.19528*, 2025a.

Rushuai Yang, Hangxing Wei, Ran Zhang, Zhiyuan Feng, Xiaoyu Chen, Tong Li, Chuheng Zhang, Li Zhao, Jiang Bian, Xiu Su, et al. Beyond human demonstrations: Diffusion-based reinforcement learning to generate data for vla training. *arXiv preprint arXiv:2509.19752*, 2025b.

Rushuai Yang, Zhuo Han, Houlin Li, Hecheng Wang, Zhichao Wu, Rui Zhang, Zhaowei Zhang, Zihong Chen, Xiaohan Yan, Chiming Liu, et al. Vine: Taming generative control policies for reinforcement learning. *arXiv preprint arXiv:2607.10369*, 2026a.

Rushuai Yang, Hecheng Wang, Zhichao Wu, Chiming Liu, Xiaohan Yan, Xuan Du, Shuoyu Yue, Chuheng Zhang, Yunlong Wang, Yongcheng Liu, et al. Aloe: Action-level off-policy evaluation for vision-language-action model post-training. *arXiv preprint arXiv:2602.12691*, 2026b.

Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Zhang Zhang, and Hao Su. Policy decorator: Model-agnostic online refinement for large policy model. In *International Conference on Learning Representations*, volume 2025, pp. 28129–28164, 2025.

He Zhang, Ying Sun, and Hui Xiong. Goldenstart: Q-guided priors and entropy control for distilling flow policies. *arXiv preprint arXiv:2603.14245*, 2026a.

Ruoqi Zhang, Ziwei Luo, Jens Sjölund, Thomas B Schön, and Per Mattsson. Entropy-regularized diffusion policy with q-ensembles for offline reinforcement learning. *Advances in neural information processing systems*, 37:98871–98897, 2024.

Shiyuan Zhang, Weitong Zhang, and Quanquan Gu. Energy-weighted flow matching for offline reinforcement learning. In *International conference on learning representations*, volume 2025, pp. 17943–17970, 2025.

Shuyi Zhang, Yunfan Lou, Hongyang Cheng, Yichen Guo, Chuyao Fu, Yaoxu Lyu, Xiaojie Zhang, Haoran Li, Pengwei Wang, Zhongyuan Wang, et al. Force: Efficient vla reinforcement fine-tuning via value-calibrated warm-up and self-distillation. *arXiv preprint arXiv:2606.26006*, 2026b.

Yang Zhang, Chenwei Wang, Yuan Zhao, Yunfei Ge, Zhenglong Sun, Xiu Li, Chi Zhang, Chenjia Bai, Xuelong Li, et al. Align-then-steer: Adapting the vision-language action models through unified latent guidance. In *International Conference on Learning Representations*, volume 2026, pp. 14717–14749, 2026c.

Yixian Zhang, Shu-ang Yu, Tonghe Zhang, Mo Guang, Haojia Hui, Kaiwen Long, Yu Wang, Chao Yu, and Wenbo Ding. Sac flow: Sample-efficient reinforcement learning of flow-based policies via velocity-reparameterized sequential modeling. In *International Conference on Learning Representations*, volume 2026, pp. 121983–122016, 2026d.

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 Additional Method Details

A.1 Affine x-Prediction Identity

We verify the interpretation used in Section 3.2. Condition on an observation s and an action a, and draw $\epsilon \sim \mathcal{N}(0, I)$. The affine path is

$$x_\sigma = (1-\sigma)a + \sigma\epsilon, \qquad u_\sigma = \epsilon - a. \tag{9}$$

The conditional velocity target u_σ satisfies

$$x_\sigma - \sigma u_\sigma = (1-\sigma)a + \sigma\epsilon - \sigma(\epsilon - a) = a. \tag{10}$$

Thus, if the velocity field equals the conditional affine target at the sampled state, the x-prediction map in Eq. 3 recovers the clean endpoint.

The population minimizer of the conditional flow-matching squared loss is the conditional mean of its velocity target. Hence, at a state x,

$$v^\star(s, x, \sigma) = \mathbb{E}[\epsilon - a \mid s, x_\sigma = x]. \tag{11}$$

Using the affine relation in Eq. 9,

$$\begin{aligned} x - \sigma v^\star(s, x, \sigma) &= x - \sigma\mathbb{E}[\epsilon - a \mid s, x_\sigma = x] && (12)\\ &= x - \mathbb{E}[\sigma(\epsilon - a) \mid s, x_\sigma = x] && (13)\\ &= \mathbb{E}[(1-\sigma)a + \sigma\epsilon - \sigma(\epsilon - a) \mid s, x_\sigma = x] && (14)\\ &= \mathbb{E}[a \mid s, x_\sigma = x]. && (15) \end{aligned}$$

Therefore, under the population-optimal velocity field, x-prediction is the conditional mean of the clean action given the interior flow state. A finite neural velocity field only approximates this identity, and edited states are evaluated by the same learned map.

Transported edit and local approximation. Let $F_{\bar\psi}(y) = y - \sigma v_{\bar\psi}(s, y, \sigma)$ denote the x-prediction map at a fixed observation and noise level. An editor perturbation Δx changes the predicted endpoint according to

$$\begin{aligned} F_{\bar\psi}(x_\sigma + \Delta x) - F_{\bar\psi}(x_\sigma) &= \Delta x - \sigma\left[v_{\bar\psi}(s, x_\sigma + \Delta x, \sigma) - v_{\bar\psi}(s, x_\sigma, \sigma)\right] && (16)\\ &= \left[I - \sigma J_v(x_\sigma)\right]\Delta x - \sigma R_v(x_\sigma, \Delta x), && (17) \end{aligned}$$

where $J_v(x_\sigma) = \nabla_x v_{\bar\psi}(s, x_\sigma, \sigma)$ and $R_v(x_\sigma, \Delta x) = o(\|\Delta x\|)$ as $\|\Delta x\| \to 0$. Dropping the remainder gives the local relation used in Eq. 4. It also makes the gradient path explicit: the critic value depends on the editor through $\partial F_{\bar\psi}/\partial x = I - \sigma J_v$, while the flow parameters are held fixed during the editor update.

A.2 Critic Training

For Q-Transport and EXPO-FT, Equation 7 regresses every critic head on the executed replay action. Their bootstrap actions are regenerated from the successor observation; the replayed next action is not used. The implementation samples two critics without replacement from a ten-member target ensemble, takes their minimum for each candidate, and then takes the maximum over candidates. The target contains no entropy term, and terminal transitions omit the bootstrap term. The target critic and base-actor parameters are updated by exponential moving average after their respective optimizer steps. DSRL uses the same TD form but evaluates its learned initial-noise variable instead of a set of ranked action candidates. SAC-Flow uses one stochastic flow sample and includes its centered path-entropy term in the target.

For Q-Transport and EXPO-FT, let $\mathcal{A}_{t+1}$ contain the K base and K edited finite candidates generated at the successor observation. If $\mathcal{J} \subset \{1, \dots, E\}$ contains the M critics sampled for a backup, the conservative candidate value is

$$V_{\text{cand}}(s_{t+1}) = \max_{a \in \mathcal{A}_{t+1}} \min_{j \in \mathcal{J}} Q_{\bar\phi, j}(s_{t+1}, a). \tag{18}$$

Table 3: Shared hyperparameters and implementation settings. Method- and task-specific values are described below.

Setting	Value
Critic learning rate η_Q	3×10^{-4}
Editor learning rate η_ψ	3×10^{-4}
Base actor LoRA learning rate η_θ	$2.5 \times 10^{-5} \to 2.5 \times 10^{-6}$, cosine decay
Temperature learning rate η_α	3×10^{-4}
Global gradient norm	1
Batch size B	64
Replay capacity	10^5
Maximum training steps	10^5
Discount factor γ	0.98
TD horizon n	2
Critic ensemble and sampled heads	$E = 10$, $M = 2$
Critic and actor target rates	$\tau_Q = 0.005$, $\tau_\pi = 0.001$
Action dimension and chunk horizon	14, $H = 10$
Euler steps and candidates per branch	$N = 4$, $K = 8$
Edited dimensions	$\mathcal{I} = \{7, \dots, 12\}$, $d_e = 6$
Optimizer and hidden layers	AdamW, 2×256

With an n-step replay return $R_t^{(n)}$ and terminal indicator d_t, the implemented target is

$$y_t = R_t^{(n)} + \gamma^n (1 - d_t) V_{\text{cand}}(s_{t+n}), \qquad \mathcal{L}_Q = \frac{1}{EB} \sum_{j=1}^{E} \sum_{i=1}^{B} \left(Q_{\phi,j}(s_i, a_i) - y_i\right)^2. \tag{19}$$

The stop-gradient boundary is applied to y_t. The target ensemble is updated after an optimizer step by

$$\bar{\phi} \leftarrow (1 - \tau_Q)\bar{\phi} + \tau_Q \phi, \qquad \bar{\psi} \leftarrow (1 - \tau_\pi)\bar{\psi} + \tau_\pi \psi. \tag{20}$$

Editor density and temperature. For the edited coordinates, the implementation samples $u \sim \mathcal{N}(\mu, \mathrm{diag}(\exp(2\ell)))$, $z = \tanh u$, and $\Delta x = \delta z$. With $D_e = H d_e$ edited coordinates per action chunk, the corresponding log-density is

$$\log \pi_\delta(\Delta x \mid s, x_\sigma, \sigma) = \sum_{q=1}^{D_e} \left[\log \mathcal{N}(u_q; \mu_q, \exp(2\ell_q)) - \log(1 - z_q^2 + 10^{-6})\right] - D_e \log \delta. \tag{21}$$

The final term is the scale Jacobian and is shared by QT, EXPO-FT, and DSRL. Writing $\beta = \log \alpha$, the adaptive temperature update minimizes

$$\mathcal{L}_\alpha(\beta) = \exp(\beta)\ \mathrm{stopgrad}(\mathcal{H} - \bar{\mathcal{H}}), \qquad \mathcal{H} = -\mathbb{E}[\log \pi_\delta], \tag{22}$$

which preserves positivity of $\alpha = \exp(\beta)$ and drives the measured entropy toward the configured target $\bar{\mathcal{H}}$.

A.3 Model and Optimization Settings

Behavior cloning starts from a task-specific $\pi_{0.5}$ LoRA checkpoint, and online adaptation starts from the resulting BC actor with the same normalization statistics. The learner and rollout process run on separate GPUs and exchange executed transitions asynchronously (Dong et al., 2026a). The critic has ten independent heads, and the critic and editor use two hidden layers of width 256. The shared settings are listed in Table 3.

The learner accumulates update credits from completed episodes. Each episode triggers the configured number of calls to `train()` once the replay buffer contains at least B transitions. Q-Transport, EXPO-FT, DSRL, SAC-Flow, HG-DAgger, and BC use respectively $1, 2, 2, 3, 5$, and 3 calls per episode. Their critic-to-actor update ratios are $20, 20, 10, 20, --, --$; thus one call performs critic 20 plus one base-actor and one editor step for Q-Transport and EXPO-FT, critic 10 plus one editor step for DSRL, critic 20 plus one editor step for SAC-Flow, and one actor step for HG-DAgger and BC. The replay capacity and total-step limit in Table 3 are hard limits.

For Q-Transport, training samples $\sigma \in \{0.25, 0.5, 0.75\}$ uniformly and inference applies the edit in the reverse order $\{0.75, 0.5, 0.25\}$. The editor uses $\delta = 0.5$ and $\alpha_0 = 0.05$, with target entropy

-30. EXPO-FT uses $\delta = 0.05$, $\alpha_0 = 1$, and target entropy -30; DSRL uses latent scale $\delta = 1$, $\alpha_0 = 0.1$, and target entropy -30. SAC-Flow uses initial temperature 0.2, target entropy 0, noise standard deviation 0.1, and action-regularization weight 100; its noise variance per Euler step is $0.1^2/N$. Q-Transport and EXPO-FT rank their $2K$ candidates with the indicated critic aggregation in the main text. DSRL does not rank action candidates, and SAC-Flow produces one differentiable sample without candidate ranking.

The base actor is updated only from successful episodes after at least 64 successful replay transitions are available. The loss weight is one, and non-edited action dimensions are zeroed in the target action. Q-Transport and EXPO-FT train the base actor; DSRL and SAC-Flow keep it fixed. HG-DAgger and BC use actor-only updates. BC selects human-intervened chunks, whereas the online base-actor update in Q-Transport and EXPO-FT uses successful replay chunks and does not apply the BC learner's `hil_only` filter. The DAgger mixing coefficient is not implemented in the current code path.

A.4 Baseline Implementations

The locally implemented baselines use the same observation pipeline, action coordinates, and normalization statistics. The three online adaptation baselines start from the same SFT checkpoint as our methods, and the critic-based implementations share the same critic architecture and replay interface. SFT (BC) uses the native $\pi_{0.5}$ LoRA objective and evaluates the resulting actor without online updates. HG-DAgger initializes a separate actor-only learner from the SFT checkpoint and updates it only on newly collected action chunks that contain human corrections; it does not use a critic or an edit policy. EXPO-FT learns a residual at the clean-action endpoint and ranks eight base and eight edited proposals with the target critic. DSRL learns an observation-conditioned distribution over selected initial-noise coordinates while leaving the remaining coordinates Gaussian and keeping the base VLA frozen. It executes one base-policy flow rollout and does not perform candidate ranking. HIL-SERL follows its original human-in-the-loop actor–critic design and is evaluated separately from the four baselines above (Luo et al., 2024b; 2025).

B Real-Robot Experimental Details

B.1 Robot System and Observation Interface

The robot receives the same two-view RGB and proprioceptive observation interface used by the $\pi_{0.5}$ policy. Each inference request predicts a ten-step action chunk, and the rollout service replans from the latest observation. Human corrections replace the executed action segment and are stored with the resulting transition.

B.2 Tasks, Rewards, and Resets

We use a sparse binary task-completion reward. Following EXPO-FT, the success detector may be rule-based or learned from success/failure trajectories; the detector returns one only when the complete manipulation objective is met. This separates task completion from intermediate contact quality, as in the success classifiers used in EXPO-FT and Focus-Then-Contact.

The five tasks are flower insertion, LAN-cable insertion, charger plug-in, and easy and hard USB insertion. Flower insertion is successful when the flower is placed in the target receptacle. LAN-cable insertion and charger plug-in are successful when the connector is fully seated in its target port. USB insertion is successful when the USB plug is fully inserted in the target port. The easy and hard variants share the same setup and camera view; the hard variant changes only two initial object positions. A timeout, safety stop, or failed insertion is unsuccessful. Except for the two USB tasks, episode initial poses receive small random perturbations around the nominal setup. USB episodes use the fixed initial setup. Resets restore the object and connector arrangement before the next episode; human intervention is permitted during online collection and is not used in autonomous evaluation.

Figure 5 shows representative camera views for the four task families. The experiments cover both open placement and precise contact-rich insertion. The easy and hard USB tasks share the same camera view; the hard variant changes only two initial object positions. Typical unsuccessful executions are summarized in Figure 6; these include misalignment, partial insertion, and loss of the manipulated object before completion.

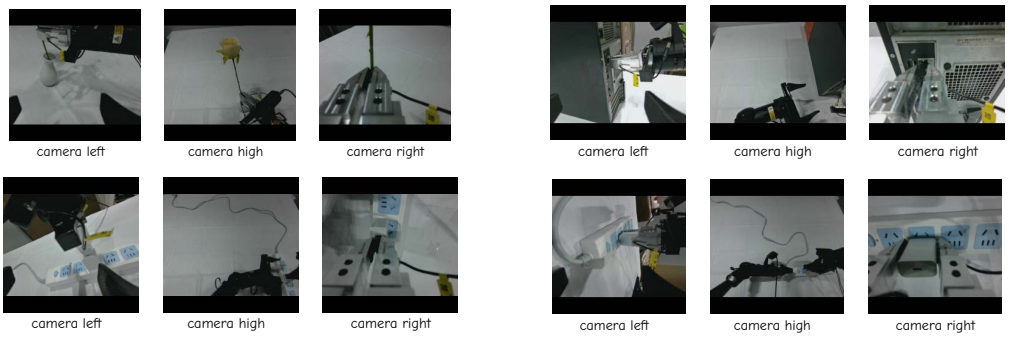


Figure 5: Representative camera views for the real-robot tasks. The USB image is shared by the easy and hard variants; the hard variant differs only in two initial object positions.

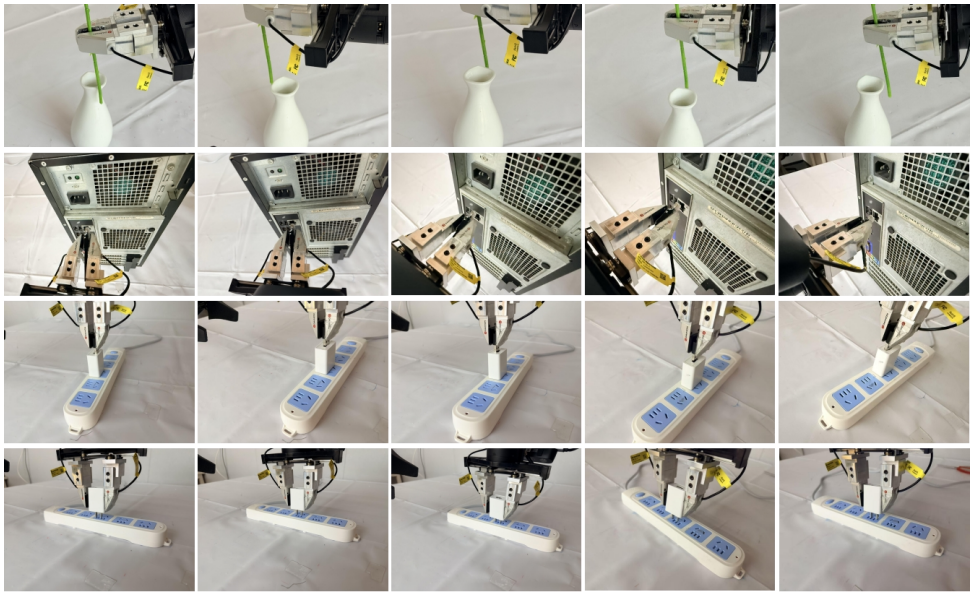


Figure 6: Representative failure modes observed during autonomous task execution.

Table 4: SFT and BC data-collection durations by task. SFT duration is the raw frame duration; BC duration is the HIL data-collection time, computed from rollout control frames at 20 Hz. USB-easy uses the shared SFT model duration.

Task	SFT time (min)	BC time (min, 20 Hz)
Flower insertion	7.88	2.28
USB insertion (easy)	7.88	13.25
USB insertion (hard)	10.79	18.25
LAN-cable insertion	10.17	19.52
Charger plug-in	10.10	14.49

Each task starts from 25 offline demonstration episodes. SFT and online BC use the task-specific demonstrations and human-intervened chunks described in Section 4. The same completion detector is used for replay success labels and autonomous evaluation.

B.3 Online Training and Evaluation Protocol

Online training begins from the task-specific BC checkpoint and continues until the configured interaction or learner-step budget is reached. Policy publications are consumed by the rollout service at episode boundaries. The learning curves use the executed control steps as the horizontal axis. Final evaluation uses the latest selected checkpoint for each method and task.

Evaluation trials are autonomous and are excluded from replay. Success is reported over 50 trials per task; timeouts and safety stops count as failures.

B.4 SFT AND BC DATA-COLLECTION DURATIONS

Table 4 summarizes the collection time used to establish the SFT and online-BC reference policies. The SFT time is the raw frame duration of the offline human-demonstration data used to initialize BC. The BC time is the human-in-the-loop (HIL) collection time during online supervised learning, computed from rollout control frames at 20 Hz; only segments in which the human intervenes contribute to the BC updates. These durations describe data collection only and exclude reset, waiting, and learner-compute time. USB insertion (easy) reuses the shared SFT model and therefore reports the same SFT duration as that model.

C ADDITIONAL RESULTS AND ABLATIONS

The supplementary results report per-task learning curves, intervention rates, and final autonomous success. The ablations isolate learned edits and the three-level interior-noise schedule on hard USB insertion and LAN-cable insertion. All ablation runs use the same BC initialization, candidate count, interaction budget, critic schedule, and successful-action base-policy update as the full method.