DRIVE: DENSE REWARD-INFORMED ACTION-VALUE LEARNING FROM MIXED EXPERIENCE FOR FLOW-BASED VLA POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) policies are predominantly trained through imitation learning on demonstration data, while the rich execution signals contained in autonomous failures and human interventions remain underexploited. Such mixed execution experience contains both effective and low-value actions, making direct imitation unsuitable for policy refinement. We propose DRIVE, a dense reward-informed action-value learning framework for refining flow-based VLA policies from mixed robot experience. DRIVE first adapts a task-conditioned visual-language progress estimator on successful trajectories, then uses it to quantify advancement, stagnation, and regression across expert demonstrations, autonomous successes, autonomous failures, and successful human-intervention trajectories. The resulting dense supervision is distilled into an action-chunk critic that estimates long-horizon values for candidate action chunks. Under the same task condition, DRIVE compares candidate values and converts their relative differences into soft refinement labels. These labels guide a negative-aware forward flow-matching objective that reinforces high-value action chunks while explicitly suppressing inferior alternatives. Importantly, the progress estimator and critic are used only during training, so the refined policy incurs no additional inference-time computation. Experiments on LIBERO and four real-world manipulation tasks show consistent improvements across multiple flow-based VLA backbones, achieving 98.25% average success on LIBERO and 62.5% on real-world tasks, with a 40.0-percentage-point improvement over the corresponding base policy.

1 INTRODUCTION

Vision-Language-Action (VLA) models have recently demonstrated strong generalization in robotic manipulation by leveraging large-scale vision-language pretraining and robot demonstrations (Zitkovich et al., 2023; Kim et al., 2024; Black et al., 2026). However, most VLA policies are trained primarily through imitation learning on successful demonstrations, which treats demonstrated actions as positive targets and remains vulnerable to distribution shift during deployment (Zhao et al., 2023; Chi et al., 2023). Meanwhile, policy deployment naturally produces diverse execution experience, including autonomous successes, stagnation, regression, failures, and human-assisted recovery, providing valuable supervision beyond additional expert demonstrations. Nevertheless, actions within such mixed trajectories differ substantially in quality: uniformly imitating them may reinforce undesirable behaviors, whereas discarding unsuccessful or corrected trajectories wastes informative signals about failure modes and recovery. The central challenge in VLA post-training is therefore how to convert mixed execution experience into reliable and discriminative action-level supervision.

A natural direction is to convert robot trajectories into denser evaluation signals beyond sparse task outcomes. Recent progress in video-language reward modeling enables dense, language-conditioned task-progress estimation from visual trajectories without manually designed rewards (Alakuijala et al., 2024; Zhai et al., 2025; Tan et al., 2025). However, these signals primarily characterize how task states evolve over time rather than the long-horizon value of executing a specific candidate action chunk under the current condition. Although action-conditioned value guidance has begun to address this limitation (Nakamoto et al., 2025), existing approaches typically rely on sparse re-

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.

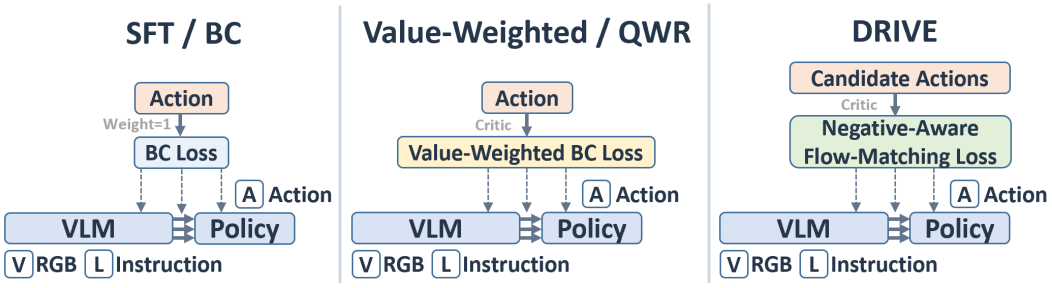


Figure 1: **Comparison of policy refinement objectives.** SFT/BC uniformly imitates observed actions; value-weighted methods such as QWR emphasize higher-value actions through weighted regression; DRIVE evaluates candidate action chunks with a critic and performs negative-aware flow-matching refinement, reinforcing high-value actions while suppressing low-value alternatives.

wards or limited execution coverage. Consequently, dense progress supervision has yet to be fully translated into reliable long-horizon action-chunk values.

Recent VLA post-training methods have begun to incorporate autonomous rollouts and human interventions beyond static expert demonstrations (Luo et al., 2024; 2025; Chen et al., 2025; Intelligence et al., 2025). While these approaches broaden the sources of training experience, they do not explicitly derive fine-grained long-horizon values for candidate action chunks. This limitation is particularly relevant to flow-based VLA policies, which model continuous action chunks through iterative velocity fields (Black et al., 2025). Standard flow matching treats each training action as a target to reproduce and provides no direct mechanism for suppressing inferior alternatives. Recent reinforcement learning methods optimize the iterative generation process (Chen et al., 2026), but are often coupled with specific sampling and stochastic optimization procedures. The remaining challenge is therefore how to extract action values from mixed experience and inject them directly into flow matching, reinforcing high-value actions while explicitly suppressing low-value alternatives without modifying inference.

To address these limitations, we propose **DRIVE**, an action-value-guided post-training framework for flow-based VLA policies. DRIVE converts dense progress supervision from mixed robot experience into action-conditioned long-horizon values, and uses these values to compare multiple action-chunk candidates under the same condition. The resulting relative supervision guides negative-aware flow refinement, reinforcing high-value actions while suppressing inferior alternatives without changing the policy architecture or inference procedure. Figure 1 contrasts DRIVE with SFT/BC and value-weighted regression (Kozakowski et al., 2021), highlighting its explicit suppression of low-value action chunks.

The contributions of our work are summarized as follows:

- We learn long-horizon action-chunk values from dense progress supervision over mixed robot experience.
- We use relative candidate values for negative-aware flow refinement, reinforcing high-value actions while suppressing low-value alternatives without changing inference.
- Experiments on LIBERO and four real-world tasks show consistent gains across multiple flow-based VLA backbones, reaching 98.25% and 62.5% average success, respectively.

2 Related Works

Dense Reward-Guided Policy Learning. Dense reward models provide richer supervision than sparse task outcomes. Language-image value representations learn goal-conditioned rewards from action-free video-language data (Ma et al., 2023), while pretrained video-language models derive rewards from a single video or language demonstration (Sontakke et al., 2023). Transferable video-language critics (Alakuijala et al., 2024), pairwise progress models (Zhai et al., 2025), and step-aware multi-view estimators (Tan et al., 2025) further provide fine-grained feedback for robot learn-

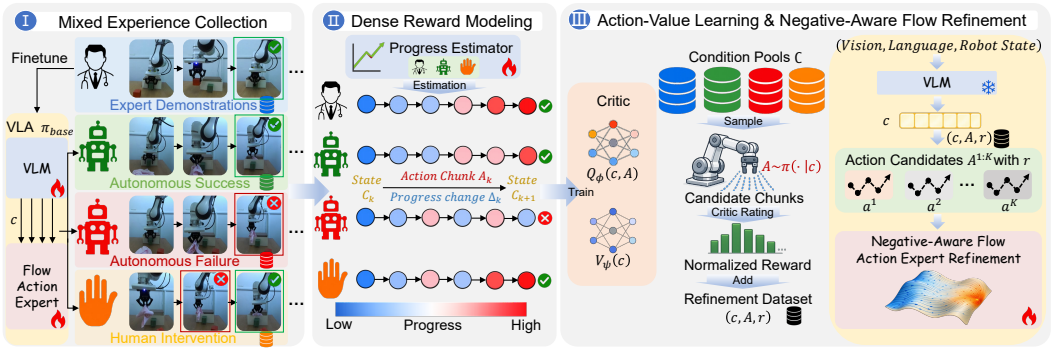


Figure 2: **Overview of DRIVE. (I)** A base flow-based VLA policy is fine-tuned on expert demonstrations and deployed to collect mixed robot experience. **(II)** A visual-language progress estimator provides dense chunk-level rewards for learning action-chunk and state values. **(III)** Logged and sampled action chunks are compared under the same condition, and their relative critic values guide negative-aware flow refinement to reinforce high-value actions and suppress inferior alternatives. Auxiliary reward and value models are used only during training.

ing. However, these approaches mainly evaluate state or trajectory progress without explicitly modeling the value of executing specific actions. Beyond reward modeling, action-conditioned critics have been explored to evaluate candidate actions for value-based selection (Nakamoto et al., 2025), but their value estimation typically relies on sparse task rewards and limited supervision. In contrast, our method distills dense progress supervision from mixed robot experience into long-horizon action-chunk values for flow-based VLA refinement.

Learning from Mixed Robot Experience. Real-world robot learning increasingly leverages mixed robot experience beyond expert demonstrations. Sample-efficient reinforcement learning combines demonstrations with online interaction (Luo et al., 2024), while human-in-the-loop extensions incorporate corrective actions during execution (Luo et al., 2025). Interactive VLA refinement further combines offline demonstrations with online corrections (Chen et al., 2025), and RECAP integrates demonstrations, autonomous trials, and expert corrections (Intelligence et al., 2025). Task-specific reinforcement learning can also generate improved trajectories for distillation into generalist robot policies (Xu et al., 2024). Meanwhile, flow-matching action experts model continuous action chunks (Black et al., 2025), with recent reinforcement learning methods optimizing their iterative generation processes (Chen et al., 2026). Our method instead extracts action values from mixed experience and incorporates them into the forward flow-matching objective without additional inference-time evaluation.

3 Preliminaries

Flow-Based VLA Policies. At chunk-level decision step k, the policy receives a visual observation o_k, a language instruction ℓ, and a robot state s_k, which are encoded and fused into the multimodal condition $c_k = E_\xi(o_k, \ell, s_k)$. Conditioned on c_k, the flow-based action expert generates an action chunk $A_k = (a_{k,0}, \ldots, a_{k,H-1}) \in \mathbb{R}^{H \times d_a}$. Given noise $\epsilon \sim \mathcal{N}(0, I)$ and flow time $\rho \sim \mathcal{U}(0, 1)$, flow matching constructs the interpolated action $A_k^\rho = (1-\rho)\epsilon + \rho A_k$ with target velocity $u_k = A_k - \epsilon$, and trains $v_\theta(c_k, A_k^\rho, \rho)$ using standard flow-matching objective $\mathcal{L}_{\text{FM}} = \mathbb{E}[\|v_\theta(c_k, A_k^\rho, \rho) - u_k\|_2^2]$.

Dense Reward-Based Chunk-Level Value Learning. Executing A_k leads to the next condition c_{k+1}. A visual-language progress estimator assigns the executed chunk a dense progress reward $\Delta_k^{\text{prog}} = R_\eta(o_k, o_{k+1} \mid \ell, o_0, o_M) \in [-1, 1]$, yielding the chunk-level transition $(c_k, A_k, \Delta_k^{\text{prog}}, c_{k+1}, d_k)$, where $d_k \in \{0, 1\}$ denotes episode termination. We use an action-chunk critic $Q_\phi(c_k, A_k)$ and a state value function $V_\psi(c_k)$, where Q_ϕ estimates the long-horizon utility induced by dense progress rewards from the mixed-experience data.

4 METHOD

We propose **DRIVE**, a dense reward-informed action-value learning framework for refining flow-based VLA policies from mixed robot experience. DRIVE follows three stages: (1) estimating dense progress rewards over mixed experience, (2) learning long-horizon action-chunk values, and (3) refining the flow-based action expert using critic-derived relative values. The progress estimator and critic are used only during training. Algorithm 1 summarizes the main training flow, while the complete procedure is provided in Appendix A.

4.1 DENSE REWARDS FROM MIXED EXPERIENCE

We first obtain a base policy $\pi_{\text{base}} = (E_\xi^{\text{base}}, v_\theta^{\text{base}})$ by supervised fine-tuning on expert demonstrations $\mathcal{D}_E$. The base policy is then executed to collect autonomous successes $\mathcal{D}_B^+$, autonomous failures $\mathcal{D}_B^-$, and successful human intervention trajectories $\mathcal{D}_H^+$. Together, these data form the mixed-experience dataset $\mathcal{D}_{\text{mix}} = \mathcal{D}_E \cup \mathcal{D}_B^+ \cup \mathcal{D}_B^- \cup \mathcal{D}_H^+$.

We initialize a visual-language progress estimator R_η from a pretrained process reward model (Tan et al., 2025) and adapt it using only successful trajectories, $\mathcal{D}_{\text{succ}} = \mathcal{D}_E \cup \mathcal{D}_B^+ \cup \mathcal{D}_H^+$. Given two observations o_p and o_q, the language instruction ℓ, and task-specific initial and goal observations o_0 and o_M, the estimator predicts relative task progress $R_\eta(o_p, o_q \mid \ell, o_0, o_M) \in [-1, 1]$. Positive values indicate task advancement, negative values indicate regression, and values near zero indicate limited progress.

After adaptation, R_η labels all trajectories in $\mathcal{D}_{\text{mix}}$, including failed executions. For each chunk-level transition (c_k, A_k, c_{k+1}, d_k), the dense progress reward is defined as

$$\Delta_k^{\text{prog}} = R_\eta(o_k, o_{k+1} \mid \ell, o_0, o_M), \tag{1}$$

For failed trajectories, o_M is obtained from a successful reference trajectory of the same task rather than the failed terminal observation. This yields the progress-labeled dataset $\mathcal{D}_Q = \{(c_k, A_k, \Delta_k^{\text{prog}}, c_{k+1}, d_k)\}$.

4.2 DENSE REWARD-INFORMED ACTION-CHUNK VALUE LEARNING

The progress estimator evaluates an action chunk through its observed state transition, but cannot score a newly generated chunk before execution. We therefore distill dense transition-level rewards into an action-chunk critic $Q_\phi(c, A)$, which estimates the long-horizon value of executing a candidate chunk under the current condition.

Given the fused multimodal condition c_k, an action encoder maps the complete chunk to $z_k = f_A(A_k)$. The critic and state value function are parameterized as $Q_\phi(c_k, A_k) = \text{MLP}_Q([c_k, z_k])$ and $V_\psi(c_k) = \text{MLP}_V(c_k)$, respectively. We train them using a chunk-level implicit Q-learning objective (Kostrikov et al., 2021). The temporal-difference target is

$$y_k = \Delta_k^{\text{prog}} + \gamma(1 - d_k)V_{\bar{\psi}}(c_{k+1}), \tag{2}$$

and the critic is optimized by

$$\mathcal{L}_Q = \mathbb{E}_{\mathcal{D}_Q}\left[\left(Q_\phi(c_k, A_k) - y_k\right)^2\right]. \tag{3}$$

The state value function is learned through expectile regression:

$$\mathcal{L}_V = \mathbb{E}_{\mathcal{D}_Q}\left[L_2^\tau\left(Q_{\bar{\phi}}(c_k, A_k) - V_\psi(c_k)\right)\right], \tag{4}$$

where $L_2^\tau(u) = |\tau - 1(u < 0)|u^2$. The target networks are updated by EMA. After training, Q_ϕ is used as a training-time evaluator for candidate action chunks during policy refinement.

4.3 MIXED-SOURCE CANDIDATE SAMPLING AND VALUATION

We construct four condition pools, $\mathcal{C}_E$, $\mathcal{C}_B^+$, $\mathcal{C}_B^-$, and $\mathcal{C}_H^+$, by extracting chunk-level decision conditions from $\mathcal{D}_E$, $\mathcal{D}_B^+$, $\mathcal{D}_B^-$, and $\mathcal{D}_H^+$, respectively. Failure trajectories provide recoverable hard

conditions at the onset of persistent stagnation or regression. Successful conditions are sampled across task-progress stages, with emphasis on later and locally inefficient segments. Intervention conditions are selected around human takeover and subsequent recovery, while expert conditions are retained across task stages for policy stabilization. During refinement, we sample a source $s \sim \mathrm{Cat}(p_{\mathrm{src}})$, where $p_{\mathrm{src}} = (p_E, p_B^+, p_B^-, p_H^+)$, $p_s \geq 0$, $\sum_s p_s = 1$, and then sample $c_i \sim \mathrm{Unif}(\mathcal{C}_s)$ from the corresponding condition pool.

For each selected condition c_i, we retain its logged action chunk A_i^1 and additionally sample $K-1$ alternative chunks from the reference policy:

$$A_i^j \sim \pi_{\mathrm{ref}}(\cdot \mid c_i), \qquad j = 2, \ldots, K. \tag{5}$$

The logged chunk and the newly sampled alternatives form the candidate group $\mathcal{G}_i = \{A_i^1, \ldots, A_i^K\}$ under the same condition. The critic then evaluates each candidate as $q_i^j = Q_\phi(c_i, A_i^j)$. Since raw critic values are not directly comparable across different conditions, we normalize the candidate values within each group and obtain the refinement label:

$$r_i^j = 0.5 + 0.5 \cdot \mathrm{clip}\left(\frac{q_i^j - \bar{q}_i}{Z_i}, -1, 1\right), \tag{6}$$

where $\bar{q}_i$ is the group mean and $Z_i = \mathrm{Std}(\{q_i^j\}_{j=1}^K) + \epsilon_Z$ is the stabilized group standard deviation, with $\epsilon_Z > 0$. The resulting labels form $\mathcal{D}_{\mathrm{ref}}$.

4.4 Negative-Aware Forward Flow Refinement

Given the critic-labeled dataset $\mathcal{D}_{\mathrm{ref}} = \{(c_i, A_i^j, r_i^j)\}$, we adapt negative-aware fine-tuning (Zheng et al., 2026) to flow-based action generation. During refinement, the condition encoder and critic are frozen, while only the action expert v_θ is optimized with an EMA reference expert $v_{\bar{\theta}}$.

For each tuple, let $v_i^j = v_\theta(c_i, A_i^{j,\rho}, \rho)$ and $v_i^{j,\mathrm{ref}} = v_{\bar{\theta}}(c_i, A_i^{j,\rho}, \rho)$, where $A_i^{j,\rho}$ and u_i^j are defined in Sec. 3. The positive and negative branches are

$$v_i^{j,\pm} = v_i^{j,\mathrm{ref}} \pm \beta\left(v_i^j - v_i^{j,\mathrm{ref}}\right), \tag{7}$$

where β controls the refinement strength. We define

$$\ell_i^{j,\pm} = \left\| v_i^{j,\pm} - u_i^j \right\|_2^2, \qquad \ell_i^j = r_i^j \ell_i^{j,+} + (1 - r_i^j)\ell_i^{j,-}, \tag{8}$$

and optimize

$$\mathcal{L}_{\mathrm{DRIVE}} = \mathbb{E}_{(c_i, A_i^j, r_i^j) \sim \mathcal{D}_{\mathrm{ref}}}\left[\ell_i^j\right]. \tag{9}$$

The role of r_i^j is characterized by the pointwise optimum of the objective:

$$v_i^{j,*} = v_i^{j,\mathrm{ref}} + \frac{2r_i^j - 1}{\beta}\left(u_i^j - v_i^{j,\mathrm{ref}}\right). \tag{10}$$

Thus, $r_i^j > 0.5$ moves the prediction toward the flow target, $r_i^j < 0.5$ pushes it away, and $r_i^j = 0.5$ preserves the reference prediction. The full derivation is provided in Appendix B.

After each iteration, the EMA reference is updated by

$$\bar{\theta} \leftarrow \lambda\bar{\theta} + (1 - \lambda)\theta, \tag{11}$$

followed by candidate resampling. At inference, only the refined VLA policy is retained; the progress estimator, critic, and candidate evaluation are discarded.

5 Experiments

We evaluate DRIVE from three perspectives: comparison with existing VLA policies on LIBERO (Section 5.1), the effectiveness of exploiting mixed experience for fine-grained real-world manipulation (Sections 5.2, 5.3), and the contribution of key components and design choices (Section 5.4).

Algorithm 1 DRIVE Training Overview

Require: $\mathcal{D}_E, \mathcal{D}_B^+, \mathcal{D}_B^-, \mathcal{D}_H^+, R_\eta, \pi_{\text{base}}$
Ensure: π_{DRIVE}
1: $\mathcal{D}_{\text{succ}} \leftarrow \mathcal{D}_E \cup \mathcal{D}_B^+ \cup \mathcal{D}_H^+, \quad \mathcal{D}_{\text{mix}} \leftarrow \mathcal{D}_{\text{succ}} \cup \mathcal{D}_B^-$.
2: Adapt R_η on $\mathcal{D}_{\text{succ}}$ and construct $\mathcal{D}_Q$ from $\mathcal{D}_{\text{mix}}$ using Eq. (1).
3: Train Q_ϕ, V_ψ on $\mathcal{D}_Q$ using Eqs. (3)–(4); freeze Q_ϕ, V_ψ.
4: Construct $\{\mathcal{C}_E, \mathcal{C}_B^+, \mathcal{C}_B^-, \mathcal{C}_H^+\}$.
5: **for** each refinement iteration **do**
6: $\quad s \sim \text{Cat}(p_{\text{src}}), c_i \sim \text{Unif}(\mathcal{C}_s); \quad \mathcal{G}_i = \{A_i^1, \ldots, A_i^K\}$ via Eq. (5).
7: $\quad q_i^j \leftarrow Q_\phi(c_i, A_i^j), \quad r_i^j \leftarrow$ Eq. (6).
8: $\quad$ Update v_θ with $\mathcal{L}_{\text{DRIVE}}$ in Eq. (9).
9: $\quad \bar{\theta} \leftarrow \lambda\bar{\theta} + (1-\lambda)\theta$.
10: **end for**
11: **return** $\pi_{\text{DRIVE}} = (E_\xi^{\text{base}}, v_\theta)$.

Table 1: **Simulation benchmark results on LIBERO.** We report mean success rate (%) ± standard deviation across three evaluation seeds. **Bold*** and underline indicate the best and second-best mean results, respectively.

Method	Spatial	Object	Goal	Long	Avg.
OpenVLA	84.7	88.4	79.2	53.7	76.5
π_0	96.8	98.8	95.8	85.2	94.2
π_0-FAST	96.4	96.8	88.6	60.2	85.5
$\pi_{0.5}$	<u>98.8</u>	98.2	**98.0***	92.4	96.8
GR00T N1	94.4	97.6	93.0	90.6	93.9
Evo-1	92.7	97.7	96.3	92.3	94.8
MolmoAct2	97.8	**100.0***	<u>97.8</u>	<u>93.2</u>	<u>97.2</u>
DRIVE	**99.00*** ±1.00	<u>99.67</u> ±0.58	**98.00*** ±1.00	**96.33*** ±1.53	**98.25*** ±0.43

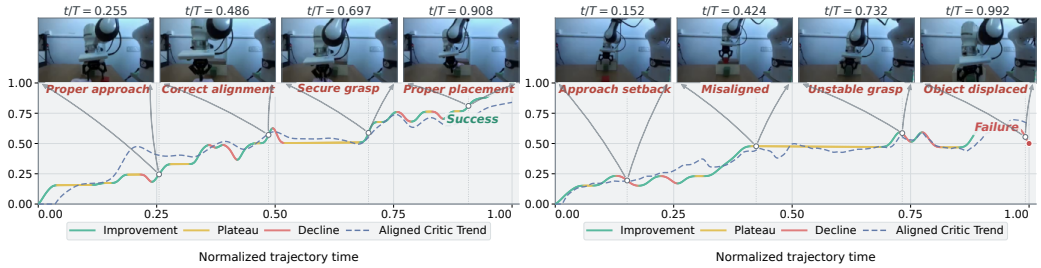
5.1 Performance on Simulation

Simulation Setup and Evaluation. We evaluate DRIVE on LIBERO (Liu et al., 2023), which comprises four suites: LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long. We compare with OpenVLA (Kim et al., 2024), π_0 (Black et al., 2026), π_0-FAST (Pertsch et al., 2025), $\pi_{0.5}$ (Black et al., 2025), GR00T N1 (NVIDIA et al., 2025), Evo-1 (Lin et al., 2025), and MolmoAct2 (Fang et al., 2026), from which DRIVE is initialized unless noted otherwise. We report success rate (%) for each suite and their average. DRIVE uses $K = 11$, $\gamma = 0.95$, $\beta = 0.5$, an EMA coefficient of 0.995, and 10 refinement iterations with 300 updates each. Table 1 reports published baseline results and DRIVE results over three evaluation seeds. Complete implementation details and per-seed results are provided in Appendices C.2 and C.4.

5.2 Real-World Experimental Setup

Setup and Tasks. We evaluate DRIVE on a Franka Research 3 platform with wrist and external RGB-D observations, using teleoperation to collect human demonstrations and interventions. We consider four real-world tasks: **Block Stack**, **Drawer Place**, **RAM Remove/Insert**, and **Plug Insert**, jointly trained in a single language-conditioned policy. Hardware, control, task descriptions, and illustrations are provided in Appendices D.1 and D.2.

Mixed Experience and Condition Sampling. For each task, we fine-tune the pretrained VLA policy with 40 expert demonstrations to obtain π_{base}, then collect 15 autonomous successes, 25 autonomous failures, and 20 successful human-intervention trajectories, yielding $40E + 15B^+ + 25B^- + 20H = 100$ trajectories per task. Human takeover is triggered near impending failure, and only eventually successful intervention episodes are retained. During refinement, we sample expert conditions across full demonstrations, success conditions from later-stage progress, failure conditions from stagnation or regression regions, and intervention conditions around takeover. Sampling uses normalized progress from R_η; full criteria and thresholds are provided in Appendix D.3.



(a) Successful trajectory **(b) Failed trajectory**

Figure 3: **Qualitative consistency of progress and learned action-chunk values on Drawer Place.** The solid curve denotes normalized global task progress, color-coded by local improvement, plateau, or decline; the dashed curve shows the critic trend after sign reversal, temporal smoothing, and trajectory-wise affine alignment, used only for visualization.

Table 2: **Overall real-world performance under a matched trajectory budget.** Success rates (%) are reported over 20 autonomous trials per task. Baseline policies are fine-tuned with 100 expert demonstrations, whereas DRIVE starts from 40 expert demonstrations and uses 100 trajectories in total through mixed-experience refinement. R/I denotes removal and insertion. The best results are marked in **bold**, and the second-best results are underlined.

Method	Block Stack	Drawer Place	RAM R/I	Plug Insert	Avg.
OpenVLA	40.0	10.0	5.0	10.0	16.3
π_0	60.0	30.0	10.0	15.0	28.8
π_0-FAST	50.0	20.0	5.0	5.0	20.0
$\pi_{0.5}$	75.0	40.0	15.0	20.0	37.5
Evo-1	40.0	25.0	5.0	10.0	20.0
MolmoAct2	80.0	35.0	10.0	15.0	35.0
DRIVE (Ours)	**95.0**	**75.0**	**35.0**	**45.0**	**62.5**

5.3 Performance on Real-World

Progress and Critic Trend Consistency. Figure 3 jointly examines the progress estimator and learned action-chunk critic on **Drawer Place**. Panels (a)–(b) show representative successful and failed trajectories, where the solid curve denotes normalized global task progress accumulated from the local progress estimates used for critic learning. Although R_η is adapted only on successful trajectories, the failed rollout exhibits clear plateaus and regressions, suggesting that the estimator remains responsive to meaningful state changes rather than simply tracking trajectory time. Since $Q_\phi(c_t, A_t)$ estimates discounted future progress rather than progress already achieved, its raw values are not directly comparable with global progress; for visualization only, we therefore apply sign reversal, temporal smoothing, and trajectory-wise affine alignment.

We further evaluate the frozen critic on 24 new autonomous failure rollouts not used during critic training. Using the original unsmoothed sequences without affine alignment, we compare global progress p_t with $Q_\phi(c_0, A_0) - Q_\phi(c_t, A_t)$. The mean per-trajectory Pearson and Spearman correlations are 0.763 ± 0.171 and 0.646 ± 0.263, respectively, and both correlations are positive for all 24 rollouts, indicating consistent temporal agreement between critic-derived trends and task progress on unseen failures.

Overall Real-World Performance. We compare DRIVE with representative VLA baselines on four real-world manipulation tasks in Table 2. All real-world baseline policies are fine-tuned and evaluated by us using the same camera views, action representation, control frequency, and evaluation protocol. The baseline policies are fine-tuned with 100 expert demonstrations per task. DRIVE instead starts from a MolmoAct2 policy fine-tuned with 40 expert demonstrations and further leverages 15 autonomous successes, 25 autonomous failures, and 20 successful human-intervention trajectories, resulting in the same total trajectory budget of 100 per task. Human control occupies at

Table 3: **Generalization of DRIVE across flow-based VLA backbones.** Each entry reports success rate (%) as 40E SFT / 100E SFT / **DRIVE**. Here, 40E SFT and 100E SFT denote policies fine-tuned with 40 and 100 expert demonstrations, respectively. DRIVE is initialized from the corresponding 40E SFT policy and refined with mixed experience, using 100 trajectories in total.

Backbone	Stack	Drawer	RAM R/I	Plug	Avg.
Evo-1	30/40/**70**	10/25/**60**	0/5/**15**	0/10/**25**	10.0/20.0/**42.5**
π_0	50/60/**75**	10/30/**60**	0/10/**20**	0/15/**35**	15.0/28.8/**47.5**
$\pi_{0.5}$	65/75/**95**	20/40/**80**	5/15/**30**	10/20/**50**	25.0/37.5/**63.8**
MolmoAct2	60/80/**95**	20/35/**75**	0/10/**35**	10/15/**45**	22.5/35.0/**62.5**

most half of each intervention trajectory, yielding no more than 50 demonstration-equivalent trajectories of active teleoperation. We report success rates (%) over 20 autonomous trials per task and the average across all four tasks.

As shown in Table 2, DRIVE achieves an average success rate of 62.5% and consistently outperforms the evaluated VLA baselines across all four tasks. Compared with the strongest 100E baseline, $\pi_{0.5}$, DRIVE improves the average success rate by 25.0 percentage points. Compared with the 100E MolmoAct2 baseline, the improvement is 27.5 percentage points. More importantly, DRIVE improves its actual 40E MolmoAct2 initialization from 22.5% to 62.5%, corresponding to a 40.0-percentage-point refinement gain. The gains are particularly pronounced on **RAM R/I** and **Plug Insert**, where precise alignment, contact-rich control, and recovery from execution errors are required. These results show that mixed execution experience can provide substantially stronger policy refinement than expert-only supervised fine-tuning under the same trajectory budget.

Generalization across Flow-Based VLA Policies. To evaluate whether DRIVE generalizes beyond a single backbone, we apply the proposed refinement framework to four representative flow-based VLA policies, as shown in Table 3. For each backbone, we report three settings: SFT with 40 expert demonstrations (40E SFT), SFT with 100 expert demonstrations (100E SFT), and DRIVE. The 40E SFT policy serves as the actual initialization for DRIVE, which is subsequently refined using mixed experience under the protocol in Section 5.2; the 100E SFT policy provides an expert-only reference under the same total trajectory budget.

As shown in Table 3, DRIVE consistently improves over both its 40E initialization and the corresponding 100E SFT reference across all four backbones. Relative to the 40E initialization, the average success-rate gains are 32.5, 32.5, 38.8, and 40.0 percentage points for Evo-1, π_0, $\pi_{0.5}$, and MolmoAct2, respectively. DRIVE also exceeds the corresponding 100E SFT policies by 22.5, 18.7, 26.3, and 27.5 percentage points. These results show that the proposed refinement is not tied to a specific VLA backbone and that mixed-experience action-value learning provides substantial gains beyond simply increasing the amount of expert supervision.

5.4 Ablation and Sensitivity Analysis

We analyze DRIVE through real-world component and data-source ablations, followed by hyperparameter sensitivity analysis on LIBERO.

Ablation of Refinement Objectives. We isolate the effect of the refinement objective under matched initialization and training data. As shown in Table 4, all variants start from the same 40E SFT policy, with mixed-experience variants using the same training data. Applying SFT directly to mixed experience improves average success only marginally from 10.0% to 12.5%. Value-Weighted Refinement, following value-weighted regression (Kozakowski et al., 2021), reaches 35.0%, showing that action-chunk values provide useful supervision. Full DRIVE further reaches 55.0% by reinforcing high-value candidates while explicitly suppressing low-value alternatives, demonstrating the benefit of negative-aware refinement beyond non-negative value weighting.

Ablation of Mixed-Experience Data Sources. We assess whether DRIVE's gains arise mainly from successful human interventions. Replacing 20 of 100 expert demonstrations with 20 successful intervention trajectories improves SFT from 22.5% to 30.0%, showing that recovery data are

Table 4: **Ablations on real-world tasks.** The upper block compares refinement objectives, while the lower block examines experience compositions. Results are success rates (%).

Variant	Drawer	RAM R/I	Avg.
40E SFT	20.0	0.0	10.0
Mixed-Experience SFT	20.0	5.0	12.5
Value-Weighted	50.0	20.0	35.0
DRIVE	**75.0**	**35.0**	**55.0**
100E SFT	35.0	10.0	22.5
80E + 20H SFT	45.0	15.0	30.0
DRIVE w/o H	60.0	30.0	45.0
DRIVE	**75.0**	**35.0**	**55.0**

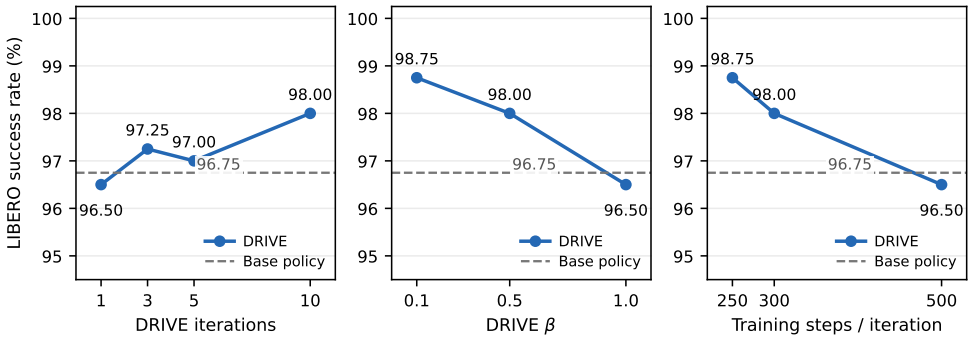


Figure 4: **Hyperparameter sensitivity of DRIVE on LIBERO.** We vary the number of refinement iterations, β, and training steps per iteration while keeping the remaining settings fixed. The dashed line denotes the base-policy success rate of 96.75%.

useful. However, DRIVE without human-intervention data already achieves 45.0%, outperforming both expert-only and intervention-augmented SFT. Adding intervention trajectories further raises DRIVE to 55.0%, indicating that recovery experience is complementary rather than the main source of improvement. The larger gain on **Drawer Place** is consistent with long-horizon action values suppressing decisions whose adverse consequences emerge later.

Hyperparameter Sensitivity on LIBERO. We further analyze three refinement hyperparameters on LIBERO by varying one at a time: the number of refinement iterations, refinement strength β, and training steps per iteration. As shown in Figure 4, repeated refinement generally improves performance, reaching 98.0% after 10 iterations. In contrast, overly aggressive updates can be detrimental: $\beta = 0.1$ and 250 steps per iteration achieve 98.75%, while $\beta = 1.0$ and 500 steps reduce performance to 96.5%. These results indicate that DRIVE benefits from iterative candidate re-evaluation with relatively conservative policy updates.

6 Conclusions

In this work, we proposed **DRIVE**, a dense reward-informed action-value learning framework for post-training flow-based VLA policies from mixed robot experience. DRIVE shows that mixed robot experience can be used more effectively when policy refinement distinguishes the long-horizon value of different action chunks instead of uniformly imitating all observed behavior. By combining dense progress-based value learning with negative-aware flow refinement, DRIVE can exploit both successful and failed experience, reinforce promising actions, and explicitly suppress low-value alternatives. This makes autonomous failures and limited human interventions useful training signals rather than data to be discarded, while retaining the original VLA architecture and inference procedure. Overall, the results suggest that action-value-guided refinement is a practical direction for improving flow-based VLA policies from heterogeneous execution experience.

AI USE STATEMENT

Generative AI tools were used during this work to assist with research ideation and methodological feedback, interpretation and organization of experimental results, literature discovery and summarization, and the drafting and language editing of parts of the manuscript. They were also used for limited assistance in preparing and reviewing experimental analysis code and mathematical exposition. All AI-assisted research ideas, technical content, code, mathematical expressions, experimental results, and cited literature were independently reviewed and verified by the authors. All experiments and reported numerical results were produced and validated by the authors. The authors take full responsibility for the final content of this work, including all text, claims, code, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies robot learning and manipulation. Human involvement is limited to providing robot demonstrations and corrective interventions through teleoperation; these trajectories are used solely as robot-control experience, and no personal or biometric information is collected or analyzed. The experiments are conducted on standard robotic manipulation platforms in a controlled research setting. We do not identify additional ethical concerns arising from the datasets, algorithms, or experimental protocols used in this work.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the DRIVE training procedure, dense reward construction, action-value learning, candidate valuation, and negative-aware flow refinement in the main paper and appendix. Algorithm 2 summarizes the complete training pipeline, while Appendix B provides the derivation of the negative-aware refinement objective. Simulation training and evaluation protocols, implementation settings, and hyperparameters are reported in Appendix C.2, with per-seed LIBERO results given in Appendix C.4. The appendix further documents the real-world robotic platform, manipulation tasks, mixed-experience construction, candidate-condition sampling, and deployment protocol.

REFERENCES

Minttu Alakuijala, Reginald McLean, Isaac Woungang, Nariman Farsad, Samuel Kaski, Pekka Marttinen, and Kai Yuan. Video-language critic: Transferable reward functions for language-conditioned robotics. *arXiv preprint arXiv:2405.19988*, 2024.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. 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. 17–40. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/black25a.html.

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, 2026. URL https://arxiv.org/abs/2410.24164.

Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Xiang Li, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, Tiejun Huang, Yu Wang, and Chao Yu. π_{RL}: Online rl fine-tuning for flow-based vision-language-action models, 2026. URL https://arxiv.org/abs/2510.25889.

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, 2025. URL https://arxiv.org/abs/2502.05450.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu (eds.), *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi: 10.15607/RSS.2023.XIX.026. URL https://doi.org/10.15607/RSS.2023.XIX.026.

Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, Shanli Xing, Jaemin Cho, Jae Sung Park, Ainaz Eftekhar, Peter Sushko, Karen Farley, Angad Wadhwa, Cole Harrison, Winson Han, Ying-Chun Lee, Eli VanderBilt, Rose Hendrix, Suveen Ellawela, Lucas Ngoo, Joyce Chai, Zhongzheng Ren, Ali Farhadi, Dieter Fox, and Ranjay Krishna. Molmoact2: Action reasoning models for real-world deployment, 2026. URL https://arxiv.org/abs/2605.02881.

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

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

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

Piotr Kozakowski, Łukasz Kaiser, Henryk Michalewski, Afroz Mohiuddin, and Katarzyna Kańska. Q-value weighted regression: Reinforcement learning with limited data, 2021. URL https://arxiv.org/abs/2102.06782.

Tao Lin, Yilei Zhong, Yuxin Du, Jingjing Zhang, Jiting Liu, Yinxinyu Chen, Encheng Gu, Ziyan Liu, Hongyi Cai, Yanwen Zou, Lixing Zou, Zhaoye Zhou, Gen Li, and Bo Zhao. Evo-1: Lightweight vision-language-action model with preserved semantic alignment, 2025. URL https://arxiv.org/abs/2511.04555.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning, 2023. URL https://arxiv.org/abs/2306.03310.

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, 2024.

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.

Yecheng Jason Ma, William Liang, Vaidehi Som, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. Liv: Language-image representations and rewards for robotic control, 2023. URL https://arxiv.org/abs/2306.00958.

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.

NVIDIA, :, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. Gr00t n1: An open foundation model for generalist humanoid robots, 2025. URL https://arxiv.org/abs/2503.14734.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models, 2025. URL https://arxiv.org/abs/2501.09747.

Sumedh A Sontakke, Jesse Zhang, Sébastien M. R. Arnold, Karl Pertsch, Erdem Bıyık, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. Roboclip: One demonstration is enough to learn robot policies, 2023. URL https://arxiv.org/abs/2310.07899.

Huajie Tan, Sixiang Chen, Yijie Xu, Zixiao Wang, Yuheng Ji, Cheng Chi, Yaoxu Lyu, Zhongxia Zhao, Xiansheng Chen, Peterson Co, et al. Robo-dopamine: General process reward modeling for high-precision robotic manipulation. *arXiv preprint arXiv:2512.23703*, 2025.

Charles Xu, Qiyang Li, Jianlan Luo, and Sergey Levine. Rldg: Robotic generalist policy distillation via reinforcement learning, 2024. URL https://arxiv.org/abs/2412.09858.

Shaopeng Zhai, Qi Zhang, Tianyi Zhang, Fuxian Huang, Haoran Zhang, Ming Zhou, Shengzhe Zhang, Litao Liu, Sixu Lin, and Jiangmiao Pang. A vision-language-action-critic model for robotic real-world reinforcement learning, 2025. URL https://arxiv.org/abs/2509.15937.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu (eds.), *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi: 10.15607/RSS.2023.XIX.016. URL https://doi.org/10.15607/RSS.2023.XIX.016.

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, 2026. URL https://arxiv.org/abs/2509.16117.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong T. Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: vision-language-action models transfer web knowledge to robotic control. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Conference on Robot Learning, CoRL 2023, 6-9 November 2023, Atlanta, GA, USA*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

A Complete DRIVE Training Procedure

Algorithm 2 DRIVE Training Pipeline

Require: Expert data $\mathcal{D}_E$, progress estimator R_η, source distribution p_{src}, candidate number K, discount γ, expectile τ, refinement strength β, EMA coefficient λ, and ϵ_Z.
Ensure: Refined policy $\pi_{\mathrm{DRIVE}} = (E_\xi^{\mathrm{base}}, v_\theta)$.

Stage I: Dense Reward and Action-Value Learning
Obtain $\pi_{\mathrm{base}} = (E_\xi^{\mathrm{base}}, v_\theta^{\mathrm{base}})$ by SFT on $\mathcal{D}_E$; roll it out to collect $\mathcal{D}_B^+$, $\mathcal{D}_B^-$, and $\mathcal{D}_H^+$.
Adapt R_η on $\mathcal{D}_{\mathrm{succ}} = \mathcal{D}_E \cup \mathcal{D}_B^+ \cup \mathcal{D}_H^+$; set $\mathcal{D}_{\mathrm{mix}} = \mathcal{D}_E \cup \mathcal{D}_B^+ \cup \mathcal{D}_B^- \cup \mathcal{D}_H^+$.
for each $(c_k, A_k, c_{k+1}, d_k) \in \mathcal{D}_{\mathrm{mix}}$ **do**
  Compute $\Delta_k^{\mathrm{prog}} = R_\eta(o_k, o_{k+1} \mid \ell, o_0, o_M)$ and store $(c_k, A_k, \Delta_k^{\mathrm{prog}}, c_{k+1}, d_k)$ in $\mathcal{D}_Q$.
end for
repeat
  Sample $(c_k, A_k, \Delta_k^{\mathrm{prog}}, c_{k+1}, d_k) \sim \mathcal{D}_Q$; $y_k = \Delta_k^{\mathrm{prog}} + \gamma(1-d_k)V_{\bar{\psi}}(c_{k+1})$.
  Update Q_ϕ with $\mathcal{L}_Q = (Q_\phi(c_k, A_k) - y_k)^2$; let $u_k = Q_{\bar{\phi}}(c_k, A_k) - V_\psi(c_k)$ and update V_ψ with $\mathcal{L}_V = |\tau - \mathbf{1}(u_k < 0)|u_k^2$.
  Update target networks $Q_{\bar{\phi}}$ and $V_{\bar{\psi}}$ by EMA.
until convergence
Freeze E_ξ^{base}, Q_ϕ, and V_ψ.
Stage II: Candidate Valuation and Flow Refinement
Construct $\mathcal{C}_E, \mathcal{C}_B^+, \mathcal{C}_B^-, \mathcal{C}_H^+$; initialize $\theta \leftarrow \theta^{\mathrm{base}}$, $\bar{\theta} \leftarrow \theta^{\mathrm{base}}$.
for each refinement iteration **do**
  Sample condition batch $\mathcal{C}$ from the four pools according to p_{src}.
  for each $c_i \in \mathcal{C}$ **do**
    Retain logged A_i^1 and sample $A_i^j \sim \pi_{\bar{\theta}}(\cdot \mid c_i)$, $j = 2, \ldots, K$.
    Compute $q_i^j = Q_\phi(c_i, A_i^j)$, $\bar{q}_i = \frac{1}{K}\sum_j q_i^j$, $Z_i = \mathrm{Std}(\{q_i^j\}) + \epsilon_Z$, and $r_i^j = 0.5 + 0.5\,\mathrm{clip}((q_i^j - \bar{q}_i)/Z_i, -1, 1)$.
    Sample $\epsilon_i^j \sim \mathcal{N}(0, I)$, $\rho \sim \mathcal{U}(0,1)$; set $A_i^{j,\rho} = (1-\rho)\epsilon_i^j + \rho A_i^j$ and $u_i^j = A_i^j - \epsilon_i^j$.
    Compute $v_i^j = v_\theta(c_i, A_i^{j,\rho}, \rho)$, $v_i^{j,\mathrm{ref}} = v_{\bar{\theta}}(c_i, A_i^{j,\rho}, \rho)$, and $v_i^{j,\pm} = v_i^{j,\mathrm{ref}} \pm \beta(v_i^j - v_i^{j,\mathrm{ref}})$.
    Compute $\ell_i^{j,\pm} = \|v_i^{j,\pm} - u_i^j\|_2^2$ and $\ell_i^j = r_i^j \ell_i^{j,+} + (1 - r_i^j)\ell_i^{j,-}$.
  end for
  Update v_θ by minimizing the batch mean of ℓ_i^j; update $\bar{\theta} \leftarrow \lambda\bar{\theta} + (1-\lambda)\theta$.
end for
return $\pi_{\mathrm{DRIVE}} = (E_\xi^{\mathrm{base}}, v_\theta)$.

B Derivation and Adaptation of Negative-Aware Flow Refinement

Our negative-aware refinement is adapted from DiffusionNFT (Zheng et al., 2026), which performs policy improvement directly through the forward diffusion/flow-matching process. DiffusionNFT constructs implicit positive and negative policies from reward-labeled samples and optimizes a single target velocity model. DRIVE adopts this principle for flow-based VLA policies, but replaces image-level rewards with action-conditioned long-horizon values estimated by the learned critic.

B.1 From Optimality Reweighting to Action-Chunk Values

Let $\pi_{\mathrm{ref}}(A \mid c)$ denote the reference policy under robot condition c, and let $r(A, c) \in [0, 1]$ denote a soft optimality label. Following the probabilistic interpretation of DiffusionNFT, the reference distribution can conceptually be separated into positive and negative components:

$$\pi^+(A \mid c) \propto r(A, c)\pi_{\mathrm{ref}}(A \mid c), \qquad \pi^-(A \mid c) \propto (1 - r(A, c))\pi_{\mathrm{ref}}(A \mid c). \tag{12}$$

Higher r therefore emphasizes the positive component, whereas lower r emphasizes the negative component.

In DiffusionNFT, r is obtained from normalized scalar rewards of generated samples. In DRIVE, the relevant supervision is instead the long-horizon value of executing an action chunk under the

same robot condition. For a candidate group $\mathcal{G}_i = \{A_i^1, \ldots, A_i^K\}$, we compute

$$q_i^j = Q_\phi(c_i, A_i^j), \qquad \bar{q}_i = \frac{1}{K}\sum_{j=1}^{K} q_i^j, \tag{13}$$

and define

$$r_i^j = \frac{1}{2} + \frac{1}{2}\,\mathrm{clip}\left(\frac{q_i^j - \bar{q}_i}{\mathrm{Std}(\{q_i^j\}_{j=1}^K) + \epsilon_Z}, -1, 1\right). \tag{14}$$

Thus, $r_i^j > 0.5$ identifies a candidate whose critic value is above the same-condition group mean, while $r_i^j < 0.5$ indicates an inferior alternative. This same-condition normalization removes much of the state-dependent value scale and converts critic scores into the soft optimality signal required by negative-aware refinement.

B.2 Adaptation to Flow-Based Action Generation

For an action chunk A_i^j, we sample $\epsilon_i^j \sim \mathcal{N}(0, I)$ and flow time $\rho \sim \mathcal{U}(0, 1)$, and construct

$$A_i^{j,\rho} = (1-\rho)\epsilon_i^j + \rho A_i^j, \qquad u_i^j = A_i^j - \epsilon_i^j. \tag{15}$$

Let

$$v_i^j = v_\theta(c_i, A_i^{j,\rho}, \rho), \qquad v_i^{j,\mathrm{ref}} = v_{\bar{\theta}}(c_i, A_i^{j,\rho}, \rho) \tag{16}$$

denote the trainable and EMA-reference velocity predictions, respectively.

Following the implicit positive/negative parameterization of DiffusionNFT (Zheng et al., 2026), DRIVE defines

$$v_i^{j,+} = (1-\beta)v_i^{j,\mathrm{ref}} + \beta v_i^j = v_i^{j,\mathrm{ref}} + \beta(v_i^j - v_i^{j,\mathrm{ref}}), \tag{17}$$

$$v_i^{j,-} = (1+\beta)v_i^{j,\mathrm{ref}} - \beta v_i^j = v_i^{j,\mathrm{ref}} - \beta(v_i^j - v_i^{j,\mathrm{ref}}), \tag{18}$$

where $\beta > 0$ controls the refinement strength. The corresponding forward flow-matching losses are

$$\ell_i^{j,+} = \|v_i^{j,+} - u_i^j\|_2^2, \qquad \ell_i^{j,-} = \|v_i^{j,-} - u_i^j\|_2^2, \tag{19}$$

and the soft negative-aware objective is

$$\ell_i^j = r_i^j \ell_i^{j,+} + (1 - r_i^j)\ell_i^{j,-}. \tag{20}$$

B.3 Why the Negative Branch Suppresses Inferior Actions

The effect of the objective can be seen directly in velocity space. For notational simplicity, define

$$d = v_\theta - v_{\mathrm{ref}}, \qquad e = v_{\mathrm{ref}} - u.$$

For one training tuple, the loss becomes

$$\ell = r\|e + \beta d\|_2^2 + (1-r)\|e - \beta d\|_2^2 \tag{21}$$

$$= \|e\|_2^2 + \beta^2\|d\|_2^2 + 2\beta(2r-1)e^\top d. \tag{22}$$

Equivalently,

$$\ell = \beta^2 \left\| d + \frac{2r-1}{\beta} e \right\|_2^2 + \left[1 - (2r-1)^2\right] \|e\|_2^2. \tag{23}$$

The point-wise optimum in velocity space is therefore

$$v_\theta^* = v_{\mathrm{ref}} + \frac{2r-1}{\beta}(u - v_{\mathrm{ref}}). \tag{24}$$

This expression gives a direct interpretation of the soft label:

$$\begin{cases} r > 0.5: & v_\theta^* \text{ moves toward the flow target } u, \\ r = 0.5: & v_\theta^* = v_{\mathrm{ref}}, \\ r < 0.5: & v_\theta^* \text{ moves away from the flow target } u. \end{cases} \tag{25}$$

Table 5: Adaptation of DiffusionNFT to flow-based VLA refinement.

Component	DiffusionNFT	DRIVE
Condition	Prompt c	Robot condition c_i
Clean sample	Image x_0	Action chunk A_i^j
Quality signal	Scalar reward	$Q_\phi(c_i, A_i^j)$
Normalization	Same-prompt group	Same-condition group
Soft label	Optimality r	Relative-value r_i^j
Reference	Sampling policy	EMA action policy
Forward target	Diffusion/flow target	Action flow target u_i^j
Optimization	Positive/negative NFT	Negative-aware flow refinement

Hence, high-value action chunks are reinforced, while low-value chunks produce an update in the opposite direction rather than merely receiving a smaller imitation weight.

This distinction separates DRIVE from conventional value-weighted flow matching. A non-negative weighted objective

$$w_i^j \|v_\theta - u_i^j\|_2^2, \qquad w_i^j \geq 0,$$

can reduce the influence of an inferior action, but its gradient still points toward that action whenever $w_i^j > 0$. In contrast, the negative branch of DRIVE reverses the refinement direction when $r_i^j < 0.5$.

B.4 Relation to DiffusionNFT

The adaptation from DiffusionNFT to DRIVE can be summarized as follows:

The key adaptation is therefore not to apply DiffusionNFT directly to robot trajectories. DRIVE first converts dense progress signals from mixed robot experience into an action-conditioned long-horizon critic. The critic then evaluates multiple candidate action chunks under an identical condition, and their relative values replace the image-level reward used by DiffusionNFT. This enables negative-aware refinement to operate at the action-generation level while preserving the original VLA inference procedure.

C Simulation Experimental Details

C.1 Evaluation Protocol

We evaluate DRIVE on the four standard LIBERO suites: LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long, each containing 10 tasks. For every task, we evaluate 10 episodes under each evaluation seed, resulting in 100 episodes per suite and 400 episodes per seed.

We use three evaluation seeds, 1000, 2000, and 3000, corresponding to 1,200 evaluation episodes in total. All three evaluations use the same trained DRIVE checkpoint; the reported standard deviation therefore reflects variation across evaluation initializations and rollout stochasticity rather than independently trained models.

An episode is considered successful when the LIBERO environment reports task completion before the evaluation horizon. The maximum episode lengths are 280 steps for LIBERO-Spatial and LIBERO-Object, 300 steps for LIBERO-Goal, and 520 steps for LIBERO-Long. We execute 50 no-op steps after each reset before policy execution.

C.2 Implementation Details

DRIVE is initialized from MolmoAct2 (Fang et al., 2026). The visual-language backbone remains frozen during DRIVE refinement, while the flow-based action expert is updated.

For critic training, we use the complete LIBERO training collection containing 1,693 episodes, 273,465 frames, and 40 tasks. Progress supervision is generated offline over 10-step action chunks. For policy refinement, we use a task-balanced subset containing 10 demonstrations per task, resulting in 400 trajectories and 66,878 frames.

Table 6: **Main LIBERO implementation hyperparameters.**

Parameter	Value
Image resolution	256×256
Action representation	Relative end-effector pose + gripper
Action dimension	7
Action-chunk horizon	10
Flow-matching steps	8
Candidate number K	11
Candidate sampling steps	15
Critic training steps	20,000
Critic learning rate	5×10^{-5}
Critic weight decay	10^{-4}
Discount factor γ	0.95
Critic global batch size	32
DRIVE iterations	10
Updates per iteration	300
DRIVE learning rate	2×10^{-6}
DRIVE weight decay	10^{-4}
Refinement coefficient β	0.5
EMA coefficient	0.995
DRIVE global batch size	56
Training GPUs	8

Table 7: **Per-seed LIBERO results for DRIVE.** Each entry reports successful episodes out of 100, with the corresponding success rate in parentheses. The same trained checkpoint is evaluated under all three seeds.

Seed	Spatial	Object	Goal	Long	Overall
1000	100/100 (100.0)	99/100 (99.0)	98/100 (98.0)	95/100 (95.0)	392/400 (98.00)
2000	99/100 (99.0)	100/100 (100.0)	97/100 (97.0)	96/100 (96.0)	392/400 (98.00)
3000	98/100 (98.0)	100/100 (100.0)	99/100 (99.0)	98/100 (98.0)	395/400 (98.75)
Mean $\pm$ Std.	99.00 ± 1.00	99.67 ± 0.58	98.00 ± 1.00	96.33 ± 1.53	98.25 ± 0.43

Table 6 summarizes the main implementation settings.

The policy receives the language instruction, agent-view RGB image, wrist-view RGB image, and robot proprioception. It predicts 10-step continuous action chunks, all of which are executed before the next policy query. During evaluation, inference uses `bfloat16` automatic mixed precision with batch size one.

C.3 Inference-Time Computation

The progress estimator and action-value critic are used only during training. At inference time, DRIVE directly generates action chunks using the refined VLA policy without progress estimation, candidate enumeration, critic evaluation, or test-time reranking. Therefore, DRIVE does not introduce additional critic or reward-model computation during deployment.

C.4 Per-Seed Results

Table 7 reports the results of the three evaluation seeds used to compute the aggregate performance in Table 1. Each suite contains 100 evaluation episodes per seed.

Across the three evaluation seeds, DRIVE succeeds in 1,179 of 1,200 rollouts, achieving an overall success rate of $98.25 \pm 0.43\%$.

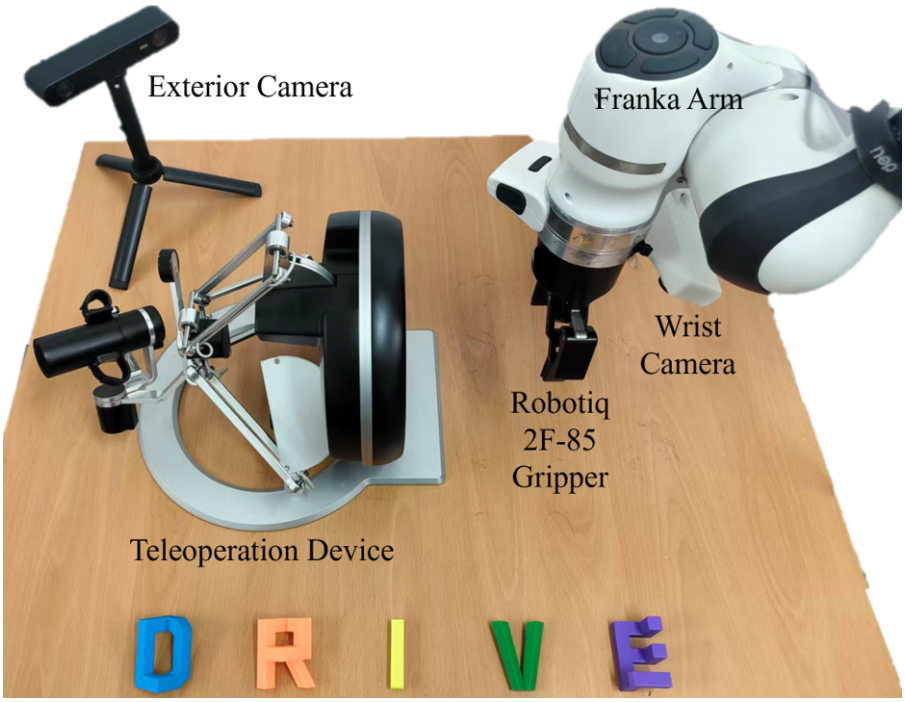


Figure 5: Real-world experimental setup used for policy training, mixed-experience collection, and autonomous evaluation.

D Real-World Experimental Details

D.1 Real-World Experimental Setup

Our real-world experiments are conducted on a 7-DoF Franka Research 3 robot arm equipped with a Robotiq 2F-85 gripper. Visual observations are provided by two RGB-D cameras: a wrist-mounted camera for close-range manipulation and an external camera overlooking the workspace. Policy inference and robot control are executed on a workstation equipped with an NVIDIA 4090 GPU.

Expert demonstrations and human interventions are collected through teleoperation. During evaluation, all policies use the same camera views, robot-state representation, action representation, control frequency, and success criteria to ensure a consistent experimental protocol. The complete real-world platform is shown in Figure 5.

D.2 Real-World Manipulation Tasks

We evaluate DRIVE on four real-world manipulation tasks covering long-horizon execution, precise alignment, and contact-rich interaction. All tasks are jointly trained in a single language-conditioned policy using pooled multi-task data, with each task specified by its corresponding language instruction. Figure 6 provides an overview of the four task setups. For each task, we additionally visualize a representative successful expert teleoperation trajectory using synchronized external and wrist-camera observations.

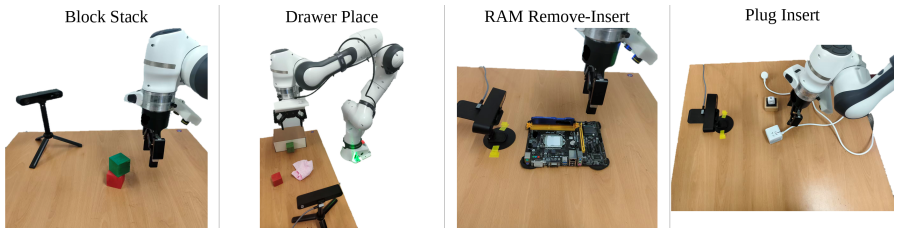


Figure 6: Overview of the four real-world manipulation tasks: Block Stack, Drawer Place, RAM Remove/Insert, and Plug Insert.

Block Stack. The robot grasps one block and places it stably on top of another block. The task requires reliable object localization, grasping, transport, and precise placement. A representative execution sequence is shown in Figure 7.

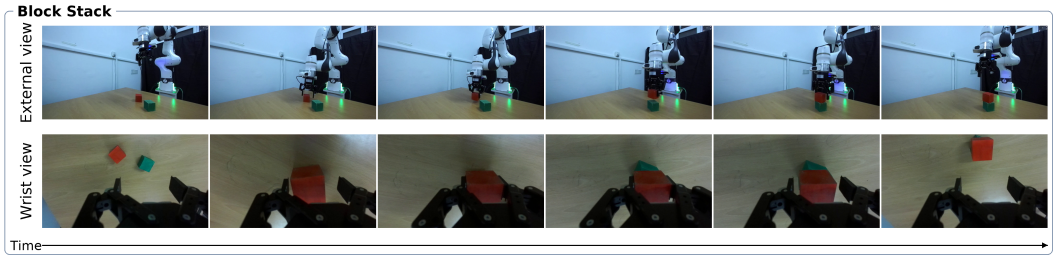


Figure 7: Representative expert trajectory for **Block Stack**. The first row shows the external-camera observations and the second row shows the synchronized wrist-camera observations, with time progressing from left to right.

Drawer Place. The robot opens a drawer, grasps an object, places it inside, and subsequently closes the drawer. The task consists of multiple dependent manipulation stages and evaluates long-horizon execution and recovery across sequential sub-actions. Figure 8 shows a representative successful trajectory.

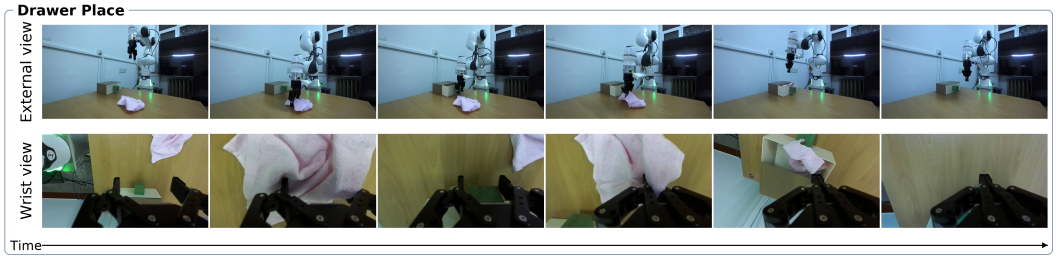


Figure 8: Representative expert trajectory for **Drawer Place**. The first and second rows show synchronized external- and wrist-camera observations, respectively, ordered chronologically from left to right.

RAM Remove/Insert. The robot removes a RAM module from one motherboard slot and inserts it into another slot. This task requires accurate grasping, fine pose alignment, and contact-rich insertion, making small execution errors particularly consequential. A representative execution sequence is shown in Figure 9.

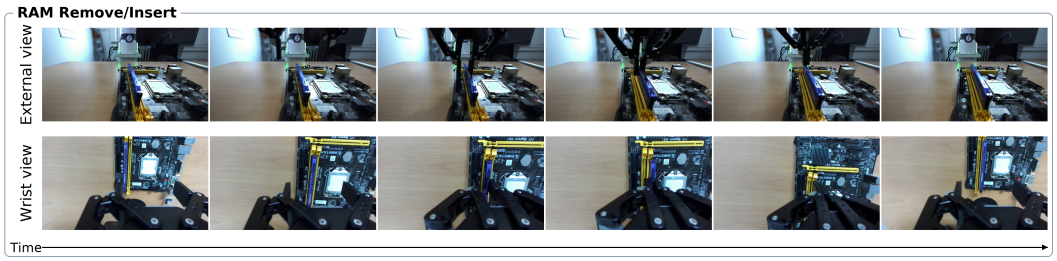


Figure 9: Representative expert trajectory for **RAM Remove/Insert**. The first row shows the external-camera observations and the second row shows the synchronized wrist-camera observations, with time progressing from left to right.

Plug Insert. The robot grasps a plug, aligns it with the target socket, and completes the insertion. Successful execution requires precise relative-pose alignment and stable contact during the final insertion stage. Figure 10 shows a representative successful trajectory.

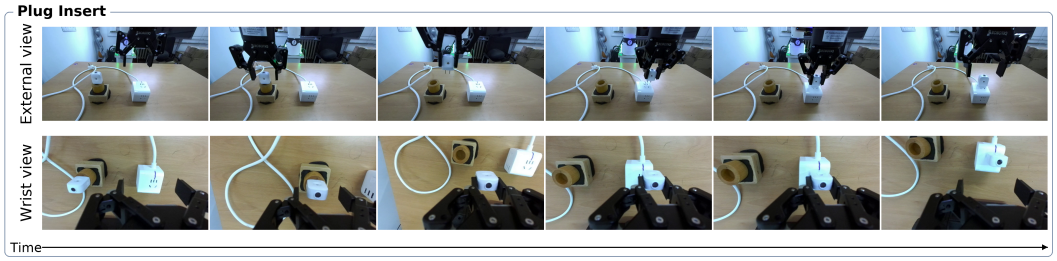


Figure 10: Representative expert trajectory for **Plug Insert**. The first and second rows show synchronized external- and wrist-camera observations, respectively, ordered chronologically from left to right.

D.3 Candidate-Condition Sampling Details

During DRIVE refinement, candidate conditions are constructed from four source-specific pools: $\mathcal{C}_E$, $\mathcal{C}_B^+$, $\mathcal{C}_B^-$, and $\mathcal{C}_H^+$, corresponding to expert demonstrations, autonomous successes, autonomous failures, and successful human-intervention trajectories, respectively.

Global Progress. For each trajectory, the adapted progress estimator predicts the local chunk-level progress

$$\Delta_k^{\text{prog}} = R_\eta(o_k, o_{k+1} \mid \ell, o_0, o_M) \in [-1, 1].$$

We accumulate and normalize these local estimates to obtain global task progress:

$$\tilde{p}_k = \sum_{t=0}^{k-1} \Delta_t^{\text{prog}}, \qquad p_k = \text{Norm}_{[0,1]}(\tilde{p}_k).$$

Here, Δ_k^{prog} is used as the dense reward for critic learning, whereas p_k is used only to construct the condition pools.

Expert Demonstrations. Expert conditions $\mathcal{C}_E$ are sampled uniformly over complete demonstrations, providing coverage across all task stages.

Autonomous Successes. For each successful rollout, we define

$$k^{50} = \min\{k : p_k \geq 0.5\},$$

and include all conditions from k^{50} to the end of the rollout in $\mathcal{C}_B^+$, emphasizing later task stages where fine-grained execution is critical.

Autonomous Failures. For each failed rollout, we identify the earliest persistent stagnation or regression point k^{stag}, using a 10% progress tolerance. All subsequent conditions are included in $\mathcal{C}_B^-$. The tolerance reduces sensitivity to small fluctuations in the progress estimate.

Human-Intervention Trajectories. For each successful human-intervention trajectory, let k^{take} and k^{release} denote the start and end of human takeover. With trajectory length T, we define

$$k^{\text{start}} = \max\big(0,\, k^{\text{take}} - \lfloor 0.1T \rfloor\big)\,,$$

and include conditions from k^{start} to k^{release} in $\mathcal{C}_H^+$, covering both the pre-takeover region and the subsequent recovery.

Source Sampling. At each refinement iteration, we sample

$$s \sim \text{Cat}(p_{\text{src}}), \qquad p_{\text{src}} = (p_E, p_B^+, p_B^-, p_H^+),$$

followed by

$$c_i \sim \text{Unif}(\mathcal{C}_s).$$

For each selected condition, DRIVE retains the logged action chunk and samples additional candidates from the reference policy for critic evaluation.

E Real-Robot Inference and Impedance Control

E.1 Multimodal Observations and Action Space

We deploy the policy on a 7-DoF Franka Research 3 (FR3) robot equipped with a parallel-jaw gripper. At each policy query, the model receives RGB images from a static third-person camera and a wrist-mounted camera, a natural-language task instruction, and the robot proprioceptive state. Both RGB streams are resized to 320×180 pixels.

The proprioceptive state is

$$\mathbf{s}_t = [\mathbf{q}_t^\top,\, g_t]^\top \in \mathbb{R}^8, \tag{26}$$

where $\mathbf{q}_t \in \mathbb{R}^7$ contains the measured joint positions and $g_t \in [0, 1]$ is the normalized gripper position.

The policy predicts chunks of absolute joint-space actions,

$$\mathbf{a}_t = [\mathbf{q}_{d,t}^\top,\, g_{d,t}]^\top \in \mathbb{R}^8, \tag{27}$$

where $\mathbf{q}_{d,t}$ and $g_{d,t}$ denote the desired joint configuration and gripper position, respectively. Joint torques are generated by the low-level impedance controller rather than directly predicted by the policy.

E.2 Control Frequencies

Policy actions are executed at $15\,\text{Hz}$, robot-side joint targets are updated at $150\,\text{Hz}$, and the Franka torque-control loop runs at $1\,\text{kHz}$. The robot-side controller interpolates consecutive desired joint positions to provide smooth high-frequency setpoints, while the desired joint velocity is set to zero during VLA execution.

During chunked inference, up to 15 predicted actions are executed before the policy is queried again, corresponding to a maximum open-loop horizon of one second.

E.3 Joint-Space Impedance Control with Jacobian-Projected Gains

Let $\mathbf{q}, \dot{\mathbf{q}} \in \mathbb{R}^7$ denote the measured joint positions and velocities, and let $\mathbf{q}_d, \dot{\mathbf{q}}_d \in \mathbb{R}^7$ denote their desired values. During VLA execution, $\dot{\mathbf{q}}_d = \mathbf{0}$. The commanded joint torque is

$$\boldsymbol{\tau} = \mathbf{K}_p(\mathbf{q})(\mathbf{q}_d - \mathbf{q}) + \mathbf{K}_d(\mathbf{q})(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) + \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}), \tag{28}$$

Table 8: Real-robot inference and impedance-control parameters.

Component	Parameter	Value
Observation	External RGB	320×180
Observation	Wrist RGB	320×180
Observation	Proprioception	7 joints + gripper
Action	Representation	Absolute joints + gripper
Policy	Action rate	15 Hz
Policy	Max. executed chunk	15 actions
Robot server	Setpoint rate	150 Hz
Torque control	Control rate	1 kHz
Controller	$\mathbf{k}_q$	$[40, 30, 50, 25, 35, 25, 10]$
Controller	$\mathbf{k}_{qd}$	$[4, 6, 5, 5, 3, 2, 1]$
Controller	$\mathbf{k}_x$	$[400, 400, 400, 15, 15, 15]$
Controller	$\mathbf{k}_{xd}$	$[37, 37, 37, 2, 2, 2]$

where $\mathbf{c}(\mathbf{q}, \dot{\mathbf{q}})$ denotes Coriolis compensation.

The configuration-dependent stiffness and damping matrices are

$$\mathbf{K}_p(\mathbf{q}) = \mathrm{Diag}(\mathbf{k}_q) + \mathbf{J}(\mathbf{q})^\top \mathrm{Diag}(\mathbf{k}_x)\mathbf{J}(\mathbf{q}), \tag{29}$$

$$\mathbf{K}_d(\mathbf{q}) = \mathrm{Diag}(\mathbf{k}_{qd}) + \mathbf{J}(\mathbf{q})^\top \mathrm{Diag}(\mathbf{k}_{xd})\mathbf{J}(\mathbf{q}), \tag{30}$$

where $\mathbf{J}(\mathbf{q}) \in \mathbb{R}^{6\times 7}$ is the end-effector Jacobian. The diagonal joint-space terms provide tracking over all seven joints, while the Jacobian-projected terms increase the effective joint-space impedance along task-space translational and rotational directions. No integral term is used.

E.4 Controller Parameters

The inference and controller parameters used in all real-robot experiments are summarized in Table 8. The first three Cartesian gain entries correspond to translation and the last three to rotation.

The same observation, action, and impedance-control configuration is used for all methods and real-robot evaluations.