EFFICIENT-RLT: SAMPLE-EFFICIENT REAL-ROBOT RL VIA OPTIMIZING HUMAN-INTERVENTION STRATEGY WITH DENSE PROCESS REWARDS

Anonymous authors
Paper under double-blind review

ABSTRACT

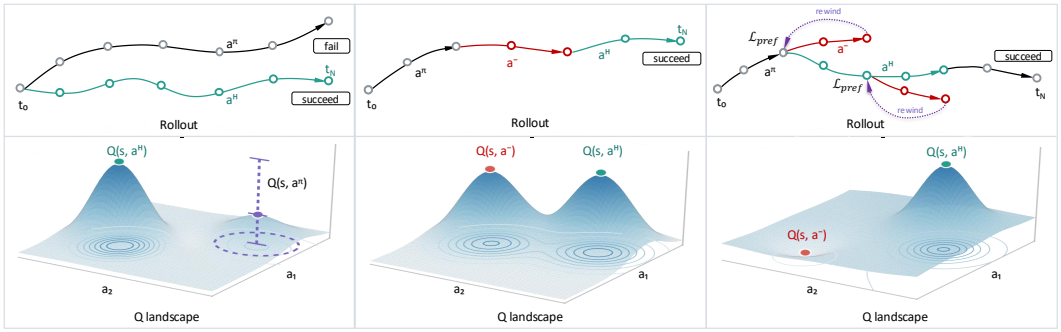
Real-world online reinforcement learning (RL) is a promising approach for post-training vision-language-action policies, particularly for precision-sensitive, long-horizon manipulation where supervised fine-tuning alone rarely achieves reliable execution. Existing human-in-the-loop methods intervene only after the policy has deviated, so the critical decision state never receives both a policy action and a corrective one. The critic therefore either extrapolates over unexecuted proposals or propagates the return of a successful recovery back through the very actions that caused the failure. We introduce Efficient RLT, a framework built on RL Token (RLT) that learns through action contrast from *rewind interventions*. When an operator flags an erroneous action chunk, the robot is physically rewound to the pre-error decision state by reverse-replaying its waypoints, and the operator issues a corrective action from there, yielding a state-aligned pair of a rejected action and its corrective replacement. Efficient RLT trains the critic and actor with a *pairwise preference learning* on these pairs, shaping a local gradient field over the action space that flows from rejected toward corrective actions, while branch-consistent credit assignment keeps the corrective return out of the rejected branch. We further train a progress estimator that supplies *dense progress rewards* under sparse task feedback, and study how takeover timing and horizon affect convergence. On Franka, CoBot, DoBot, and X-Square platforms across diverse and challenging manipulation tasks, Efficient RLT improves sample efficiency, convergence speed, and final success rate. The code will be publicly available.

1 INTRODUCTION

General-purpose vision-language-action (VLA) models and World Action Models (WAMs) (Ye et al., 2026) have demonstrated the ability to learn and execute a diverse range of manipulation tasks. However, when deployed on real-world robotic systems, they still struggle with challenging tasks that require fine-grained and precise manipulation, where even a minor deviation at a critical step can lead to failure. This issue becomes more pronounced in long-horizon tasks, as errors may compound over time and the model often lacks an explicit awareness of task progress. Reinforcement learning (RL) offers a promising post-training paradigm for addressing these challenges. By directly optimizing task outcomes through environmental interaction, RL can improve the precision and robustness of VLA models (Black et al., 2025; Xu et al., 2025; Zhang et al., 2026), particularly for manipulation skills that are difficult to acquire through demonstration-based behavior cloning alone.

To preserve the general capabilities of VLA models, reduce the action space for exploration, and adapt actions in specific states, existing real-world reinforcement learning methods commonly employ residual policies (Yuan et al., 2025) to refine the actions proposed by a pretrained VLA policy. Nevertheless, directly applying online RL to physical robots remains challenging due to the prohibitively large exploration space and sparse reward signals. Classical deep RL algorithms such as SAC learn optimal policies through extensive trial-and-error interactions and reward feedback from simulated environments. Since real-robot operation is highly time-consuming, RL training is ideally completed within a practical time budget of only a few hours. To mitigate these challenges, prior methods, such as HIL-SERL (Luo et al., 2025), RL Token (RLT) (Xu et al., 2026a), and EXPO-

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.



(a) Full Intervention (b) Delayed Intervention (c) Rewind Intervention

Figure 1: How intervention design shapes critic learning in real-world online RL. (a) Full takeover: only human actions a^H are executed, so the critic is unconstrained at the unexecuted policy proposals a^π and extrapolates their values. (b) Delayed takeover without a terminal boundary: the return of a successful correction propagates back through the rejected actions a_1^- and a_2^-, spuriously assigning them high Q-values. (c) Rewind intervention: the robot is restored to the pre-error state, so the rejected action a^- and its corrective replacement a^+ share one state. The critic is trained to rank the corrective action above the rejected one by a margin proportional to how far apart the two actions are, which makes it action-discriminative without making it arbitrarily steep.

FT (Dong et al., 2026), rely on human intervention to provide informative, high-value corrective actions during online training. These methods build on actor–critic architectures and employ either delayed or full takeover for human intervention, which can expose the policy to erroneous intermediate actions and slow convergence. This limitation is particularly detrimental to precise manipulation and multi-stage long-horizon tasks.

We propose physical action rollback mechanism: upon an erroneous action, execution is paused, the robot is physically restored to a suitable state, and a human takes over to provide a corrective action. We then contrast the erroneous and corrective branches, favoring the corrective action while suppressing the erroneous one. This comparison rapidly calibrates the Q-function while preserving necessary exploration (Figure 1 (c)), mitigating the fine-grained control bias caused by Q-function extrapolation. As a result, the actor quickly learns the correct actions.

2 Related Work

Real-World Reinforcement Learning Classic deep RL algorithms learn from large numbers of parallel trial-and-error rollouts in simulation, using binary success/failure feedback and hand-crafted dense rewards. Recent methods such as π_{RL} (Chen et al., 2025) and SimpleVLA-RL (Li et al., 2026) have integrated VLA models (e.g., π_0 (Black et al., 2024)) with RL algorithms (e.g., PPO, GRPO) for simulated robotic manipulation. On real robots, however, these assumptions break down: each interaction is slow and needs human effort, the exploration space is large, and rewards are sparse. For real-world RL, HIL-SERL extends SERL (Luo et al., 2024), a framework built on the classic off-policy SAC algorithm with lightweight policy networks, by incorporating human-in-the-loop interventions. The interventions supply corrective actions during online training, which reduces exploration time and accelerates convergence. RL Token freezes the VLA and trains a lightweight residual actor–critic on a compact token extracted from its representations, but relies solely on delayed intervention and treats human interventions only as ordinary replay data. EXPO-FT combines SFT and RL framework by pairing a pretrained flow-based VLA with a lightweight edit policy that refines its sampled actions, selecting among candidates via a learned Q-function.

Embodied Progress Reward Learned reward models for robotic manipulation increasingly replace binary episode outcomes with temporally structured progress or value signals. SARM (Chen et al., 2026) jointly predicts task stage and within-stage progress from natural-language subtask annotations, providing supervision that is less sensitive to variable demonstration length. TimeRewarder (Liu et al., 2025) learns temporal distances from passive videos and converts their frame-wise

differences into dense rewards, removing the need for action labels. Other methods formulate the problem as general value estimation. RynnValue (Huang et al., 2026) uses language-conditioned temporal distance as a scalable cost-to-go target, while World Value Models (Wang et al., 2026) combine a video world-model backbone with distributional value prediction to evaluate both expert and suboptimal trajectories. Robo-ValueRL (Xia et al., 2026) studies how history-conditioned value reliability, global task progress, and local action preference affect offline data selection and online policy improvement. Process-reward methods further address temporal context and failure semantics. Robo-Dopamine (Tan et al., 2025) constructs a multi-view, step-aware reward model and combines it with policy-invariant reward shaping. Robo-Dopamine 2.0 (Xu et al., 2026b) extends this direction with history-conditioned queries, signed progress for valid, failed, and recovery states, and transition-aware replay. PACE (Song et al., 2026) introduces a phase-progress-aware remaining-cost critic together with progressive policy distillation, whereas PRIMO R1 (Liu et al., 2026) trains a video MLLM as an active process critic using initial–current state anchoring and outcome-based reinforcement learning. Collectively, these works establish that reliable progress estimation must account for temporal context, phase structure, suboptimal behavior, and failure-aware regression. They generally introduce a separate reward or value model, however, whereas RL Token (Xu et al., 2026a) exposes a compact interface for lightweight online adaptation of a pretrained VLA. Our method builds on this interface to decode progress from the same task-conditioned representation used by the deployed policy.

3 Approach

Our method is based on the ideas of RLT and EXPO-FT multi-candidate actions. Given an observation $\mathbf{o}_t = [\mathbf{I}_t^1, \ldots, \mathbf{I}_t^n, \ell, \mathbf{q}_t]$, a generalist action model outputs an action chunk $\tilde{\mathbf{a}}_{t:t+H}$. $\mathbf{I}_t^i$ is i^{th} image, ℓ is the language instruction, and $\mathbf{q}_t$ is a vector of joint angles. An encoder–decoder module then compresses the model's internal representations into a compact token $\mathbf{z}_{rl}$. In stage 1, the model and encoder–decoder module are first fine-tuned with the demonstration datasets. We then train a progress estimator on top of $\mathbf{z}_{rl}$.

For stage 2 online RL training, the token $\mathbf{z}_{rl}$ is concatenated with $\mathbf{q}_t$ to form the RL state $\mathbf{s}_t = [\mathbf{z}_{rl}; \mathbf{q}_t]$, which serves as the input to both the actor and the critic. The actor is a residual policy $\pi_{res,\theta}$ that refines the action chunk,

$$\mathbf{a}_t = \tilde{\mathbf{a}}_t + \rho \tanh(\mathbf{a}_{t,res} + \sigma\boldsymbol{\epsilon},), \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{1}$$

where $\rho \in (0, 2]$ is the edit scale that controls the magnitude of the action modification. $\boldsymbol{\epsilon}$ is the exploration noise. Off-policy RL collects transition data (s_t, a_t, s_{t+1}, r_t) by continuously interacting with the environment, and stores the data in a replay buffer. The actor π_θ is trained to produce actions that the critic assigns a high value. Concretely, θ is updated by gradient ascent on $Q_\phi(\mathbf{s}_t, \mathbf{a})$, which amounts to minimizing the following loss:

$$\mathcal{L}_\pi(\theta) = -\,\mathbb{E}_{\mathbf{s}_t \sim \mathcal{D},\, \mathbf{a} \sim \pi_\theta(\cdot|\mathbf{s}_t)}\big[Q_\phi(\mathbf{s}_t, \mathbf{a})\big], \tag{2}$$

where $\mathcal{D}$ denotes the replay buffer and the critic parameters ϕ are held fixed during the actor update. The critic aims to accurately estimate the value of every state–action pair (s_t, a_t), and is trained with temporal-difference (TD) learning:

$$\mathcal{L}_{TD}(\phi) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}}\left[\big(Q_\phi(s_t, a_t) - y_t\big)^2\right], \quad y_t = r_t + \gamma \min_{i=1,\ldots,10} Q_{\phi_i^-}\big(s_{t+1}, \tilde{a}^*_{t+1}\big), \tag{3}$$

where $\bar{\phi}$ denotes the target network. A common failure mode in actor-critic training is Q-value overestimation, which leads the actor to learn erroneous actions. Following TD3 (Fujimoto et al., 2018), we therefore add small noise to the target action $\tilde{a}^*_{t+1} = a^*_{t+1} + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$. To accelerate training, we exploit the ability of flow-based VLA models to draw multiple samples. Following EXPO-FT, the base model generates multiple candidate actions in a single pass, and the residual policy edits each of them, yielding multiple refined candidates. The inference and TD backup policy is to select the value-maximizing candidate across base and edited actions:

$$a^* = \operatorname*{arg\,max}_{a \in \bigcup_{i=1}^N \{a_i, \tilde{a}_i\}} Q_\phi(s, a). \tag{4}$$

We follow the offline-to-online setting, where the actor and critic are first trained with Cal-QL (Nakamoto et al., 2023) on expert demonstration data to provide a good baseline for both the actor and the critic, thereby accelerating the online RL stage.

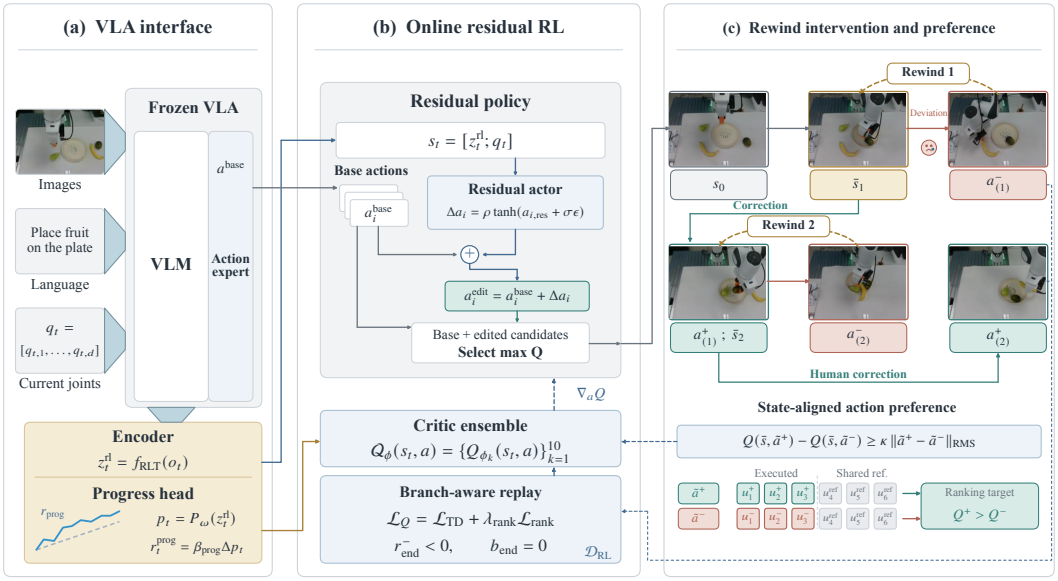


Figure 2: Overview of Efficient-RLT. (a) A frozen VLA maps multimodal observations to base actions, while an encoder extracts compact RL tokens and a progress head provides dense progress-based rewards. (b) During online residual RL, a residual actor proposes action corrections, and the critic selects among the base and edited candidates while being trained with branch-aware replay. (c) Rewind interventions return the robot to pre-error states, where human corrections yield state-aligned action preferences that train the critic to rank corrective actions above rejected ones.

3.0.1 Action Preference Learning from Rewind Intervention

Rewind intervention. We consider chunk-level control, where the policy executes an action chunk $\mathbf{a}_t \in \mathbb{R}^{C \times d_a}$ from state s_t and observes the next state s_{t+1}. When the operator judges that the recent execution has deviated from the task, the robot is not taken over at the deviated state. Instead, it is physically rewound to $\bar{s}$, the last satisfactory state before the deviation, and the operator issues a corrective continuation from there. Each rewind therefore yields two executed branches that fork from the same state: $\tau^{\pm} = \left(\bar{s}, \mathbf{a}_0^{\pm}, s_1^{\pm}, \ldots, \mathbf{a}_{K_\pm - 1}^{\pm}, s_{K_\pm}^{\pm}\right)$, where τ^- is the rejected branch executed by the policy and τ^+ is the corrective branch executed after the rewind; the two may differ in length. Since $\bar{s}$ is recorded before the first corrective action is executed, both branches are anchored to exactly the same state, and their leading actions form a state-aligned preference $Q(\bar{s}, \mathbf{a}_0^+) > Q(\bar{s}, \mathbf{a}_0^-)$ This comparison involves two actions that were both executed from $\bar{s}$ and whose consequences were both observed.

Branch-consistent credit assignment. Naively connecting a successful takeover to the preceding trajectory may propagate positive return through actions that were explicitly rejected. We instead retain both branches in replay while preserving their respective bootstrap paths. For each action-chunk transition, we use

$$y_t = R_t^{(C)} + \gamma^C (1 - d_t) b_t \min_j \bar{Q}_j \left(s_{t+1}, \hat{\mathbf{a}}_{t+1}\right), \qquad R_t^{(C)} = \sum_{c=0}^{C-1} \gamma^c r_{t,c}, \tag{5}$$

where b_t is a bootstrap mask and $\hat{\mathbf{a}}_{t+1}$ is the branch-consistent successor action.

For the rejected branch, we assign its final transition a negative reward $\rho^- < 0$ and set $b_t = 0$. Earlier rejected transitions retain $b_t = 1$ and bootstrap through the next executed rejected action, propagating the negative outcome backward to $\mathbf{a}_0^-$. In contrast, the transition ending at the rewind state $\bar{s}$ is not marked terminal; it bootstraps through the first corrective action $\mathbf{a}_0^+$. Consequently, negative credit is confined to the rejected branch, while the corrective return propagates into the shared trajectory prefix.

State-aligned preference pairs. Each rewind event adds a preference tuple

$$\zeta_i = \left(\bar{s}_i,\, \mathbf{a}_i^{\text{ref}},\, \mathbf{a}_i^{+},\, \mathbf{a}_i^{-},\, \mathbf{M}_i,\, c_i\right) \in \mathcal{D}_{\text{pref}}, \tag{6}$$

where $\mathbf{a}_i^{+}$ and $\mathbf{a}_i^{-}$ are the corrective and rejected actions at the shared rewind state, and c_i denotes preference confidence. $\mathbf{M}_i$ identifies the executed prefix of a partially completed action chunk; the remaining suffix is shared by both candidates and therefore does not affect their comparison. The VLA proposal $\mathbf{a}_i^{\text{ref}}$ provides the reference action used by the residual policy. During training, transition and preference mini-batches are sampled independently as $\mathcal{B}_{\text{RL}} \sim \mathcal{D}_{\text{RL}}$ and $\mathcal{B}_{\text{pref}} \sim \mathcal{D}_{\text{pref}}$; no cross-buffer state matching is required because each preference tuple is self-contained.

Pairwise critic preference learning. Because an action chunk may be only partially executed before the rewind, the critic evaluates

$$\widetilde{\mathbf{a}}_i^{\pm} = \mathbf{M}_i \odot \mathbf{a}_i^{\pm} + (1 - \mathbf{M}_i) \odot \mathbf{a}_i^{\text{ref}}, \tag{7}$$

so that the two candidates differ only in the executed dimensions and share the VLA reference elsewhere. A constant ranking margin is ill-suited to rewind pairs: a rewind may discard as little as a single chunk, in which case $\widetilde{\mathbf{a}}_i^{+}$ and $\widetilde{\mathbf{a}}_i^{-}$ are nearly identical and a fixed gap can only be met by making Q arbitrarily steep in action space, precisely where the actor reads $\nabla_{\mathbf{a}} Q$. We therefore scale the margin with the action gap $d_i = \|\widetilde{\mathbf{a}}_i^{+} - \widetilde{\mathbf{a}}_i^{-}\|_{\text{RMS}}$ and require, for every critic Q_{θ_j} in the ensemble,

$$Q_{\theta_j}\left(\bar{s}_i, \widetilde{\mathbf{a}}_i^{+}\right) - Q_{\theta_j}\left(\bar{s}_i, \widetilde{\mathbf{a}}_i^{-}\right) \geq \kappa\, d_i, \tag{8}$$

where $\kappa > 0$ is the required slope. Eq. equation 8 lower-bounds the average directional derivative of Q from the rejected toward the corrective action, and vanishes when the two actions coincide.

The endpoint constraint alone only bounds the secant slope; it does not guarantee that Q increases in this direction near $\widetilde{\mathbf{a}}_i^{-}$, where the policy currently acts. We thus impose the same constraint at intermediate points $\mathbf{a}_i(u) = \widetilde{\mathbf{a}}_i^{-} + u\,(\widetilde{\mathbf{a}}_i^{+} - \widetilde{\mathbf{a}}_i^{-})$ with a proportionally scaled margin $\kappa\, u\, d_i$. Writing

$$h_{ij}(u) = \left[\kappa\, u\, d_i - Q_{\theta_j}\left(\bar{s}_i, \mathbf{a}_i(u)\right) + Q_{\theta_j}\left(\bar{s}_i, \widetilde{\mathbf{a}}_i^{-}\right)\right]_+, \tag{9}$$

where $[x]_+ = \max(0, x)$, the confidence-weighted ranking loss is

$$\mathcal{L}_{\text{rank}} = \frac{1}{\sum_{i \in \mathcal{B}_{\text{pref}}} c_i} \sum_{i \in \mathcal{B}_{\text{pref}}} \frac{c_i}{N_Q} \sum_{j=1}^{N_Q} \Big(h_{ij}(1) + \frac{\lambda_{\text{loc}}}{K} \sum_{k=1}^{K} h_{ij}(u_{ik}) \Big), \qquad u_{ik} \sim \mathcal{U}(0,1), \tag{10}$$

with K probes per pair. Since Q enters linearly, the probe terms require only additional critic forward passes. The hinge becomes inactive once the required slope is attained, leaving the TD objective to determine the absolute value scale. The complete critic objective is

$$\mathcal{L}_Q = \mathcal{L}_{\text{TD}}\left(\mathcal{B}_{\text{RL}}\right) + \lambda_{\text{rank}} \mathcal{L}_{\text{rank}}\left(\mathcal{B}_{\text{pref}}\right). \tag{11}$$

Optionally, the same pairs can also be applied directly to the actor through a Bradley–Terry preference objective; details are provided in Appendix A.2.

Local critic smoothing. Preference pairs are sparse and correspond to individual physical states. To improve local robustness without perturbing robot execution, we optionally augment only the critic action input for replay transitions marked as human interventions or rejected rewind actions:

$$\mathbf{a}^{\text{aug}} = \text{clip}_{\mathcal{A}}\left(\mathbf{a} + \text{clip}(\boldsymbol{\epsilon}, -\delta, \delta)\right), \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma_{\text{aug}}^2 I). \tag{12}$$

This augmentation is used only when evaluating the TD regression loss. The executed actions stored in replay, the corrective targets used by the actor, the exact positive–negative preference pairs, and all branch-specific next-action links remain unchanged. It therefore encourages local smoothness of the learned critic without corrupting the state-aligned rewind supervision.

3.1 Progress Head Design

Our progress estimator is motivated by two observations. First, task progress provides a compact intermediate signal for long-horizon credit assignment. Second, the VLA already encodes task-conditioned visual, language, and proprioceptive information, so progress can be decoded from the same internal representation used for action generation rather than from a separate reward model.

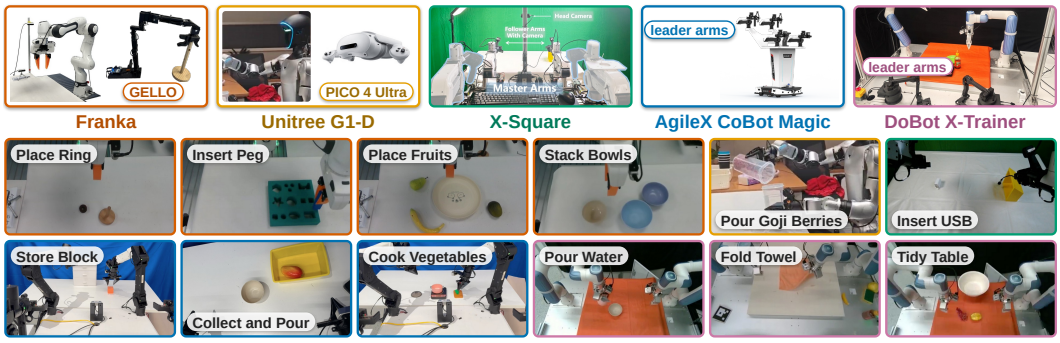


Figure 3: **Robot platforms and tasks.** Top: the five platforms used in our real-world experiments; tags mark the device through which a human operator takes over when the policy errors. Bottom: the 12 evaluation tasks, framed in the colour of the platform they run on.

We therefore build the estimator on the frozen RL Token (Xu et al., 2026a) interface, which exposes a compact readout of the pretrained VLA for sample-efficient online learning.

This choice is directly motivated by *Decoding Task Progress from VLA Representations* (Bhardwaj et al., 2026) which shows that normalized task progress is linearly readable from intermediate VLA activations and can support runtime monitoring, even without changing the policy itself. We use this observability result as a design principle: the progress signal should be decoded from the policy's own task-conditioned representation and remain an auxiliary readout rather than a second action-generating model.

Let o_t denote the observation at time t and let

$$z_t^{\mathrm{rl}} = f_{\mathrm{RLT}}(o_t) \tag{13}$$

be the frozen RL Token representation. We attach a lightweight two-layer regression head with a sigmoid output:

$$p_t = P_\omega(z_t^{\mathrm{rl}}) = \sigma\big(W_2\,\phi(W_1 z_t^{\mathrm{rl}} + b_1) + b_2\big) \in [0,1], \tag{14}$$

where only $\omega = \{W_1, b_1, W_2, b_2\}$ is trained. The scalar p_t denotes the relative completion progress, with larger values indicating states closer to successful termination. The head does not generate actions and does not replace the intervention policy; it only provides a task-progress readout.

3.2 Constructing Progress from Successful Trajectories

We first define progress on trajectories with a known successful endpoint. For either a successful teleoperation trajectory or a successful autonomous trajectory generated by the base/RLT policy, $\tau = \{s_0, \ldots, s_T\}$, we assign the normalized temporal target

$$y_t = \frac{t}{T}, \qquad t = 0, \ldots, T. \tag{15}$$

The target increases smoothly from zero at the task start to one at successful completion. Teleoperation trajectories provide a reliable initial reference, while autonomous successes expose the head to the state distribution induced by the policy being optimized. Both sources share the same progress semantics, so they train one scalar predictor rather than separate expert and policy models. The progress head is fitted with a regression loss,

$$\mathcal{L}_{\mathrm{prog}} = \mathbb{E}_{(z_t^{\mathrm{rl}}, y_t)}\left[\big(P_\omega(z_t^{\mathrm{rl}}) - y_t\big)^2\right]. \tag{16}$$

At execution time, the predicted progress difference

$$\Delta p_t = p_t - p_{t-1} \tag{17}$$

measures local advancement or regression and can be used as a dense reward component, $r_t^{\mathrm{prog}} = \beta_{\mathrm{prog}} \Delta p_t$. Persistent positive, near-zero, or negative values correspond respectively to progress, stagnation, or likely regression. Intervention-derived trajectories and their specialized labels are treated separately in the subsequent data-construction and training sections.

4 EXPERIMENTS

We evaluate Efficient-RLT on real robots to answer three questions. **(Q1)** Does learning from rewind interventions improve a VLA policy beyond RL Token, which learns from conventional takeovers? **(Q2)** How much of the gain comes from rewind intervention with action preference learning, and how much from the dense progress reward? **(Q3)** Does rewind intervention still help when the frozen base is not $\pi_{0.5}$?

4.1 EXPERIMENTAL SETUP

Platforms and intervention devices. We use five platforms with diverse morphologies (Fig. 3): a single-arm Franka, the Unitree G1-D humanoid, and the bimanual X-Square, AgileX CoBot Magic, and DoBot X-Trainer. The operator intervenes through GELLO (Wu et al., 2024) on Franka, a PICO 4 Ultra headset on G1-D, and leader arms on the three bimanual systems (Appendix A.5).

Training. For the Franka, CoBot, DoBot, and X-Square comparisons, we fine-tune $\pi_{0.5}$ and the RL token $\mathbf{z}_{rl}$ for 30,000 steps on 200 demonstrations per task, train the progress head for 20,000 steps, and pre-train the actor and critic with Cal-QL for 40,000 steps before online RL. The G1-D base-model ablation uses a different frozen base (Section **??**).

Tasks. Franka: *Place Ring* (place a ring on a rod), *Insert Peg* (insert a peg into its matching hole), *Stack Bowls* (stack bowls in size order), and *Place Fruits* (place three fruits on a plate). CoBot: *Assemble Parts* (insert one part with each gripper), *Store Block* (open a drawer, place a block inside, and close it), *Collect and Pour* (collect fruit in a bowl and pour it into a basket), and *Cook Vegetables* (put vegetables in a pot and cover it). DoBot: *Pour Water*, *Fold Towel*, and *Tidy Table*. X-Square: *Insert USB* (insert a USB connector held by one arm into a plug held by the other). G1-D: *Pouring goji berries*. The tasks combine contact-rich precision steps with multi-stage sequences (Fig. 4).

Metrics and baselines. We report success rate (SR) and a 10-point process score that sums completed milestones (Appendix A.9), which separates failed long-horizon rollouts that stop at different stages. Each $\pi_{0.5}$ comparison uses 30 rollouts per task. We compare against the SFT base $\pi_{0.5}$ (Black et al., 2025) and RL Token (Xu et al., 2026a), which trains a residual policy on the same RL token from standard takeover interventions.

4.2 MAIN RESULTS

Table 1 answers Q1. Averaged over the 12 tasks, Efficient-RLT reaches 71.1% SR and a process score of 8.6, compared with 63.9% and 7.9 for RL Token and 55.0% and 6.9 for $\pi_{0.5}$. It matches or exceeds RL Token in SR on all 12 tasks and is strictly better on 9.

Gains concentrate on precision-critical steps. The largest improvements over RL Token occur where the outcome hinges on a single fine-grained contact: *Insert Peg* (70.0%→96.7%), *Place Ring* (13.3%→30.0%), and *Assemble Parts* (10.0%→20.0%). These contacts are where operators most often intervene (Fig. 4), and therefore where rewind produces state-aligned pairs. A takeover without rewind supervises only the human action and leaves the rejected proposal unexecuted, so the critic must extrapolate at exactly this state (Fig. 1a). The slope-scaled ranking loss (Eq. 10) instead makes Q increase from $\mathbf{a}^-$ toward $\mathbf{a}^+$ where the actor acts, so the actor gradient points at the correction that a fine alignment needs.

Competent behaviour is preserved. On tasks that $\pi_{0.5}$ already solves in at least 86.7% of rollouts (*Store Block*, *Collect and Pour*, *Fold Towel*, *Tidy Table*), Efficient-RLT keeps or raises SR, and its process score stays within 0.5 points of the base. RL Token instead falls below the base on *Store Block* (96.7%→90.0%). This is consistent with branch-consistent credit assignment: the return of a successful correction is not propagated back into the actions the operator rejected (Fig. 1b), so the critic does not learn to favour them.

Remaining failures. Where SR ties RL Token (*Cook Vegetables*, *Pour Water*, *Insert USB*), the process score is equal or higher, most visibly on *Insert USB* (3.5→5.1): rollouts reach later milestones, but the final step still fails. Bimanual insertion remains the hardest setting for every method (*Insert USB*, 6.7%; *Assemble Parts*, 20.0%).

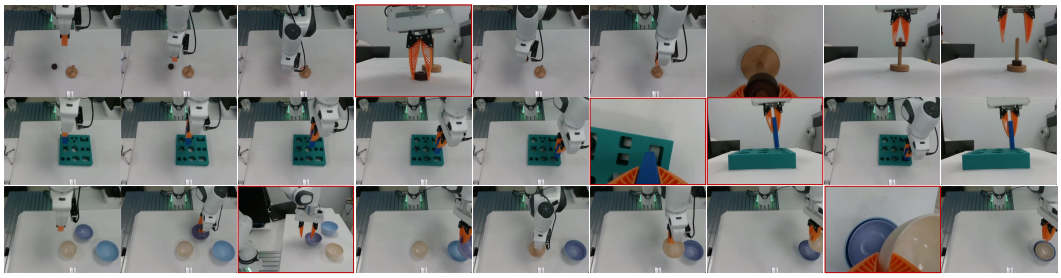


Figure 4: Illustration of the experiment task execution process. Red boxes highlight the key actions and the points that often require human intervention.

Table 1: Real-world evaluation across robot platforms. Best per-task results are bold.

Task	$\pi_{0.5}$		RL Token		**Efficient-RLT**	
	SR (%) ↑	Score /10 ↑	SR (%) ↑	Score /10 ↑	SR (%) ↑	Score /10 ↑
Franka						
Place Ring	3.3	1.1	13.3	3.6	**30.0**	**5.8**
Insert Peg	26.7	4.7	70.0	8.4	**96.7**	**9.9**
Stack Bowls	73.3	8.3	80.0	8.8	**90.0**	**9.3**
Place Fruits	80.0	9.1	86.7	9.3	**93.3**	**9.7**
CoBot						
Assemble Parts	3.3	4.3	10.0	5.1	**20.0**	**6.1**
Store Block	**96.7**	**9.8**	90.0	9.5	**96.7**	**9.8**
Collect and Pour	86.7	**9.6**	90.0	9.5	**93.3**	9.3
Cook Vegetables	46.7	7.1	**63.3**	8.8	**63.3**	**8.9**
DoBot						
Pour Water	60.0	8.9	**66.7**	**9.3**	**66.7**	**9.3**
Fold Towel	90.0	9.2	93.3	9.5	**96.7**	**9.7**
Tidy Table	93.3	9.5	96.7	9.8	**100.0**	**10.0**
X-Square						
Insert USB	0.0	1.8	**6.7**	3.5	**6.7**	**5.1**

4.3 Ablation Study

Components (Q2). Figure 5 adds the components of Efficient-RLT to RL Token one at a time on the four Franka tasks (process scores in Appendix A.3). Rewind intervention with action preference learning (RI + AP) raises the mean SR from 62.5% to 73.4%, and the dense progress reward raises it to 77.5%, so RI + AP accounts for 72% of the total gain. RI + AP helps most on *Insert Peg* (+20.0) and *Place Ring* (+13.4), the precision-critical tasks identified in Section 4.2. The progress reward adds a smaller 3.3–6.7 points per task. It credits advancement between milestones but carries no information about which action to take at the failure state, so it complements the rewind pairs rather than replacing them.

4.4 Ablating the Base Model on G1-D

Rewind on a humanoid with a different base (Q3). On G1-D *Pouring goji berries*, all variants share a frozen EWAM base, so the scores are not comparable to Table 1. Over 10 rollouts each, EWAM alone scores 2.7/10, Efficient-RLT without rewind 2.5, and full Efficient-RLT 4.9 (Appendix A.8). Without rewind, residual RL does not improve the base; it only shifts which arm succeeds, raising left-cup grasps from 3 to 7 of 10 while right-cup grasps fall from 9 to 3. With rewind, both grasps succeed (10 and 9 of 10) and the cups are brought together in 8 of 10 rollouts, so the benefit of rewind does not depend on the $\pi_{0.5}$ base. The pour itself remains rare (1 of 10 rollouts; Appendix A.4).

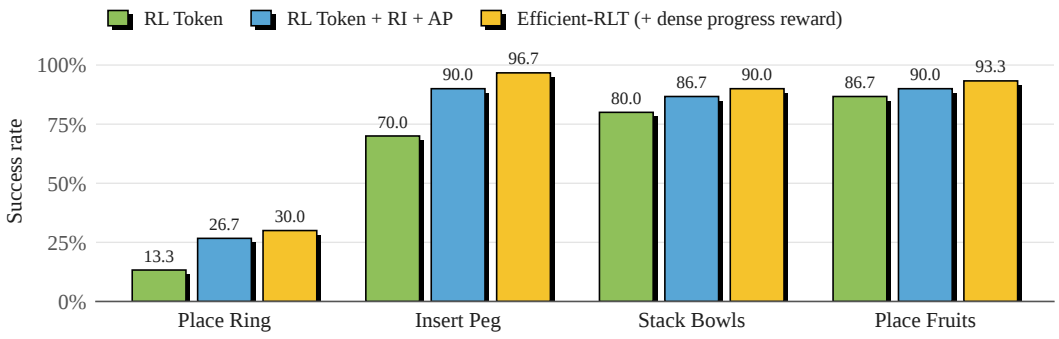


Figure 5: Component ablation on the four Franka tasks (success rate, %). RI: rewind intervention; AP: action preference learning. Efficient-RLT adds the dense progress reward on top of RL Token + RI + AP.

Table 2: Mean process score on G1-D *Pouring goji berries*, averaged over 10 rollouts. All variants share the EWAM base, not the $\pi_{0.5}$ base above. Higher is better. Best in **bold**.

	EWAM (base)	w/o rewind takeover	**Efficient-RLT (full)**
Score /10	2.7	2.5	**4.9**

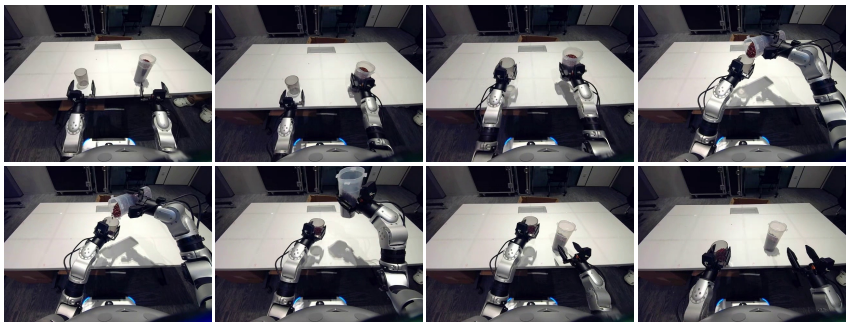


Figure 6: Execution sequence of *Pouring goji berries* on the Unitree G1-D. The robot grasps both cups, lifts them into alignment, pours the goji berries from one cup into the other, and returns both cups to the table.

5 Conclusion

We presented Efficient-RLT, which turns each human correction into supervision the critic can use. Rewinding the robot to the pre-error state yields a rejected and a corrective action from the same state; branch-consistent credit assignment keeps their returns apart, and a slope-scaled ranking loss shapes the critic so that its action gradient points from the rejected toward the corrective action. A progress head read out from the RL token adds a dense reward for long-horizon tasks. On all 12 tasks of the platform comparison, Efficient-RLT matches or exceeds RL Token, with the largest gains on precision-critical insertion and placement; the ablations attribute most of this gain to rewind-based preference learning, and on a humanoid with a different base, rewind is what allows residual RL to improve at all. A natural next step is to reduce the operator effort that rewind still requires, for example by using the progress head to detect deviations automatically. We discuss limitations in Appendix A.1.

AI USE STATEMENT

In this work, we used generative AI tools to clean and reformat datasets, assist with translation, and create and edit auxiliary software code. We also used these tools to draft and revise manuscript text for clarity, format LaTeX and experimental tables, and assist with literature searches and reference formatting. We did not use generative AI to generate synthetic datasets, design or provide feedback on our research methodology or experiments, implement the proposed Efficient-RLT method, or analyze or interpret experimental results. The research hypotheses, conceptual framework, and mathematical formulations underlying Efficient-RLT were developed by the authors, as were the reasoning and derivations supporting them.

The authors reviewed all AI-assisted work. AI-generated or AI-edited code was read, tested, and verified for correctness by the authors before use. All experimental measurements reported in this paper were obtained from physical robot runs using our own experimental pipeline. The authors checked AI-assisted text for technical accuracy and consistency with the work performed, and checked AI-assisted reference entries against the original sources for accuracy and relevance. We take responsibility for the final content of this work, including all text, claims, code, and other artifacts produced with the assistance of generative AI.

REFERENCES

Atiksh Bhardwaj, Edward Weiyi Duan, Prithwish Dan, Wei-Chiu Ma, and Preston Culbertson. Decoding task progress from vla representations. *arXiv preprint arXiv:2608.13474*, 2026.

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.

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, 2025. URL `https://proceedings.mlr.press/v305/black25a.html`.

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, 2025.

Qianzhong Chen, Justin Yu, Mac Schwager, Pieter Abbeel, Fred Shentu, and Philipp Wu. Sarm: Stage-aware reward modeling for long horizon robot manipulation. In *International Conference on Learning Representations*, volume 2026, pp. 17091–17116, 2026.

Perry Dong, Kuo-Han Hung, Tian Gao, Dorsa Sadigh, and Chelsea Finn. Expo-ft: Sample-efficient reinforcement learning finetuning for vision-language-action models. *arXiv preprint arXiv:2605.25477*, 2026.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, 2018.

Dongchi Huang, Hongyin Zhang, Bohan Hou, Siteng Huang, Zhian Su, Hang Guo, Tong Lu, Zhaofeng Xu, Jiahao Tang, Jianfei Yang, et al. Rynnvalue: Scaling robotic value foundation models with temporal distance. *arXiv preprint arXiv:2608.09853*, 2026.

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

Yibin Liu, Yaxing Lyu, Daqi Gao, Zhixuan Liang, Weiliang Tang, Shilong Mu, Xiaokang Yang, and Yao Mu. From passive observer to active critic: Reinforcement learning elicits process reasoning for robotic manipulation. *arXiv preprint arXiv:2603.15600*, 2026.

Yuyang Liu, Chuan Wen, Yihang Hu, Dinesh Jayaraman, and Yang Gao. Timerewarder: Learning dense reward from passive videos via frame-wise temporal distance. *arXiv preprint arXiv:2509.26627*, 2025.

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, 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.

Mitsuhiko Nakamoto, Yuexiang Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-QL: Calibrated offline RL pre-training for efficient online fine-tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Chengye Song, Jiawei Zhang, Rui Song, Shengqi Wang, Xiangrong Zhang, Ziyi Wang, Huanbin Zhou, and Hongzhou Wang. Pace: Phase-progress-aware credit for long-horizon embodied manipulation. *arXiv preprint arXiv:2608.15026*, 2026.

Yu Sun, Meng Cao, Yang Ping, Kaidong Zhang, Qingxuan Chen, Rongtao Xu, Liangwang Ruan, Xuecheng Chen, Dongxiu Liu, Yunxiao Yan, Zunnan Xu, Runze Xu, Charles Yang, Peilun Zhang, Xiaofan Li, Ruyi Gan, Liang Ma, Yuehao Yin, Jincheng Yu, Lufang Chen, Yuxin Liang, Peng Zhai, Hao Wang, Ivan Laptev, Ian Reid, Qian Wang, and Xiaodan Liang. Maniparena: Comprehensive real-world evaluation of reasoning-oriented generalist robot manipulation, 2026. URL https://arxiv.org/abs/2603.28545.

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.

Zhihao Wang, Jianxiong Li, Yu Cui, Yuan Gao, Xianyuan Zhan, Junzhi Yu, and Xiao Ma. World value models for robotic manipulation. *arXiv preprint arXiv:2606.24742*, 2026.

Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 12156–12163. IEEE, 2024.

Wenke Xia, Pei Ren, Wenbo Yu, Yizhuo Zhang, Jifan Li, Yixue Zhang, Yinuo Zhao, Qingyang Gao, Jianlong Fu, Jian Tang, et al. Robo-valuerl: Reliable value estimation for offline-to-online reinforcement learning. *arXiv preprint arXiv:2607.09866*, 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*, 2026a. URL https://arxiv.org/abs/2604.23073.

Rongtao Xu, Jian Zhang, Minghao Guo, Youpeng Wen, Haoting Yang, Min Lin, Jianzheng Huang, Zhe Li, Kaidong Zhang, Liqiong Wang, et al. A_0: An affordance-aware hierarchical model for general robotic manipulation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 13491–13501. IEEE, 2025.

Yijie Xu, Haopeng Jin, Run Zhou, Shengbang Liu, Sixiang Chen, Hongyang Cheng, Sicheng Hu, Peterson Co, Jinwen Luo, Huajie Tan, et al. Robo-dopamine 2.0: History-conditioned and ood-aware process reward modeling for robotic manipulation. *arXiv preprint arXiv:2608.15680*, 2026b.

Seonghyeon Ye et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

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.

Kaidong Zhang, Jian Zhang, Rongtao Xu, Yu Sun, Youpeng Wen, Shuoshuo Xue, Xiaoyu Guo, Minghao Guo, Weijia Liufu, Liu Zihou, et al. A1: Adaptive truncated vision-language-action model from affordance to action. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1503–1514, 2026.

A Appendix

A.1 Limitations

Efficient-RLT requires sustained operator supervision during online RL. Operators monitor rollouts, provide reward feedback, identify deviations, restore the system to a pre-error state, and teleoperate corrective actions. Although the progress estimator supplies a dense reward component, it does not replace these human judgments or interventions. Improvements in robot interaction efficiency therefore need not translate into proportional savings in operator time, which we do not quantify.

Rewind-based action comparisons also assume that the pre-error state can be recovered with sufficient fidelity, an assumption that may be difficult to satisfy after irreversible interactions or changes to object configurations. The remaining failures in our experiments are consistent with these limits. Bimanual insertion stays difficult for every method (*Insert USB*, 6.7% SR; *Assemble Parts*, 20.0%), and on G1-D *Pouring goji berries*, rewind repairs the bimanual approach but not the pour, which succeeds in 1 of 10 rollouts. A spilled pour cannot be undone, so rewind yields few corrective pairs at that stage.

A.2 Approach Details

Actor preference learning. Shaping the critic alone does not guarantee that the actor follows it, so we also apply each pair directly to the residual policy. Treating the actor as a fixed-variance Gaussian $\mathcal{N}(\boldsymbol{\mu}_\phi(\bar{s}_i, \mathbf{a}_i^{\text{ref}}), \sigma_\pi^2 I)$, the log-likelihood gap between the corrective and rejected actions reduces, up to cancelling constants, to a difference of masked squared distances:

$$\Delta_{\pi,i} = \frac{1}{2\sigma_\pi^2}\Big(\big\|\mathbf{M}_i \odot (\mathbf{a}_i^- - \boldsymbol{\mu}_\phi)\big\|^2 - \big\|\mathbf{M}_i \odot (\mathbf{a}_i^+ - \boldsymbol{\mu}_\phi)\big\|^2\Big), \tag{18}$$

which is positive when the actor mean lies closer to the corrective action. We increase this gap with a confidence-weighted Bradley–Terry objective

$$\mathcal{L}_{\text{pref}} = -\frac{1}{\sum_i c_i} \sum_{i \in \mathcal{B}_{\text{pref}}} c_i \log \operatorname{sigmoid}\left(\beta\, \Delta_{\pi,i}\right), \tag{19}$$

where $\beta > 0$ sets its sharpness, and add $\lambda_{\text{pref}} \mathcal{L}_{\text{pref}}$ to the base actor objective, i.e., the $-Q(s, \pi_\phi(s))$ term and any behavior-cloning regularizer of the residual policy. Both preference terms are enabled once the buffer holds enough pairs.

A.3 Component Ablation: Process Score

Figure 7 reports the process score for the component ablation of Figure 5. The ordering matches success rate: the mean over the four Franka tasks rises from 7.5 for RL Token to 8.4 with RI + AP and to 8.7 with the dense progress reward. The gap is widest on *Place Ring* (3.6, 5.2, 5.8), where most rollouts fail and partial progress separates the variants. On the other three tasks every variant scores at least 8.4, so the 10-point scale compresses the differences that success rate shows.

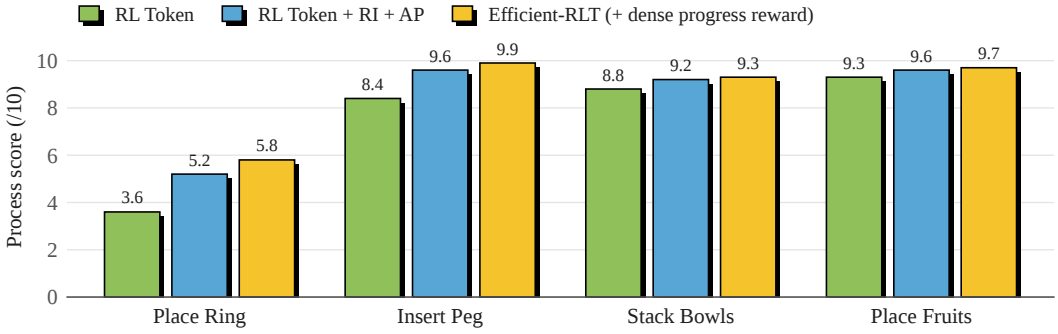


Figure 7: Component ablation on the four Franka tasks (process score, out of 10). RI: rewind intervention; AP: action preference learning. Efficient-RLT adds the dense progress reward on top of RL Token + RI + AP.

A.4 Ablating the Base Model on G1-D

The comparisons above freeze a $\pi_{0.5}$ base. The G1-D ablation uses a different one. Here the frozen policy is EWAM, our world action model, and Efficient-RLT adapts only the residual RL head. The three variants therefore share one EWAM initialization, and their scores are not comparable to the $\pi_{0.5}$ rows in Table 1. On the Unitree G1-D task *Pouring goji berries*, the robot grasps two cups,

brings them together, pours the goji berries from one cup into the other, and places both cups back on the table (Fig. 6). Table 2 compares three variants that share this EWAM initialization: (i) EWAM alone; (ii) Efficient-RLT without rewind takeover; and (iii) full Efficient-RLT. Each rollout receives the sum of its completed milestones under the rubric in Table 4. We report the mean of this process score over 10 rollouts, and do not convert the outcomes into a success rate: rollouts that stop at different stages are not equivalent. Table 2 gives these means. Appendix A.8 lists how often each milestone is completed.

The three means come from different failure points, not from a uniform gain. EWAM completes the right-cup grasp in 9 of 10 rollouts but the left-cup grasp in only 3, and it brings the cups together only in those 3. The frozen model therefore reaches one cup and stops; its score of 2.7 is a single-arm grasp, not a coordinated pour. Residual post-training without rewind does not repair this. It grasps the left cup more often, in 7 of 10 rollouts, but the right cup in only 3, reaches alignment in 3, and scores zero in 3. The update changes which arm succeeds. Without a corrective action issued from the pre-error state, it does not learn the joint approach, and the mean falls to 2.5. Full Efficient-RLT is the only variant that keeps both grasps, in 10 and 9 rollouts, and brings the cups together in 8. This comparison isolates rewind takeover: the robot is returned to the last satisfactory state, and the rejected approach is paired with a corrective continuation. That is the supervision required to grasp both cups and bring them together, and it is where the score increases. Of the 2.2 points over EWAM, 1.4 come from the left-cup grasp and 0.5 from alignment; the single successful pour adds 0.3. The shared limit is the pour itself. It occurs in 1, 0, and 0 rollouts for the full method, EWAM, and the no-rewind variant. Neither cup is placed back on the table except in one no-rewind rollout, which placed both cups without pouring. Rewind therefore accounts for the gain on the bimanual approach. It does not yet produce a reliable pour or a completed placement.

A.5 Experimental Robotic Arm and Intervention Arm

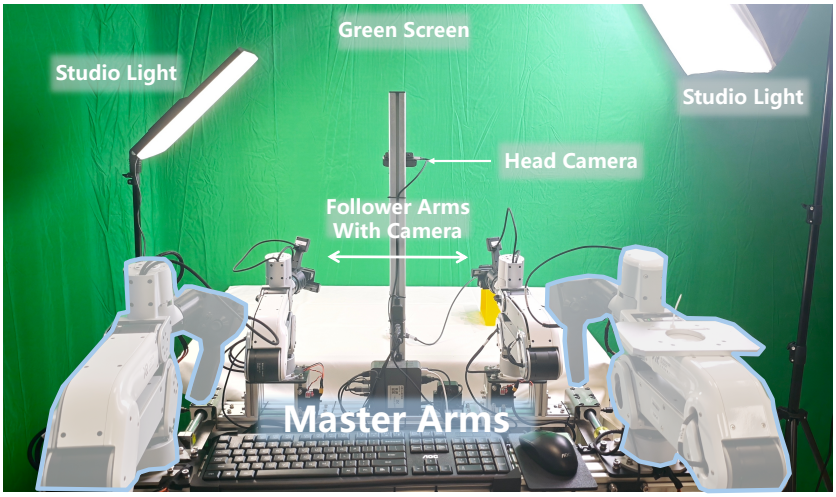


Figure 8: Overview of the tabletop experimental platform. The fixed-base bimanual system comprises two camera-equipped follower arms, two master arms for teleoperation and intervention, a centrally mounted head camera, fixed studio lighting, and a green-screen background.

As illustrated in Fig. 8, our experiments are conducted on a fixed-base bimanual tabletop platform consisting of two follower arms and a pair of master arms used as the teleoperation and intervention interface. Each follower arm is equipped with a wrist-mounted camera to capture close-range visual observations of object interactions, while a centrally mounted head camera provides a global view of the workspace. During demonstration collection, operator commands are transmitted through the master arms to control the corresponding follower arms. To reduce irrelevant visual and environmental variations, the platform is enclosed by a green screen and illuminated using fixed studio lights, with the camera placement and workspace geometry kept unchanged throughout the experiments. The system operates at 30 Hz, providing synchronized multi-view visual observations and proprioceptive signals for policy training and evaluation Sun et al. (2026).

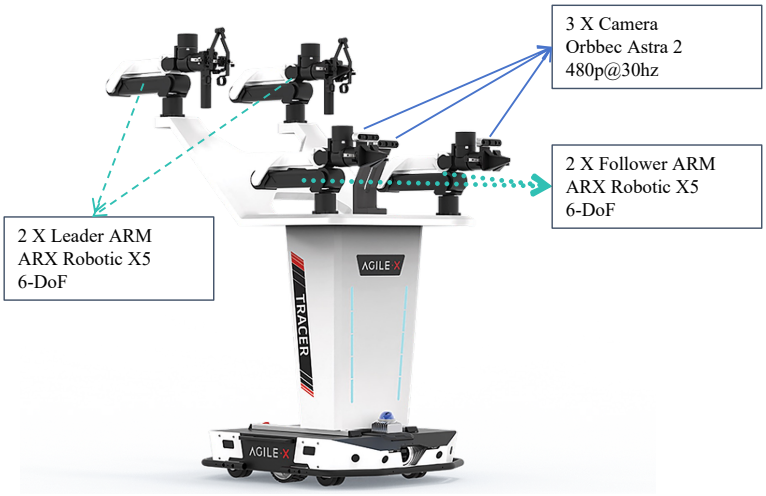


Figure 9: The AgileX CoBot Magic platform used in our experiments. The system includes two ARX Robotic X5 follower arms, two corresponding leader arms for teleoperation and intervention, and three Orbbec Astra 2 cameras.

A.6 CoBot Magic

Figure 9 shows the AgileX CoBot Magic bimanual robot platform used in our experiments. CoBot Magic is a mobile manipulation platform designed for teleoperation, robot data collection, and embodied-intelligence research. Our configuration comprises two six-degree-of-freedom ARX Robotic X5 follower arms, two corresponding ARX Robotic X5 leader arms for teleoperation and human intervention, and three Orbbec Astra 2 cameras. The cameras capture synchronized visual observations at 480p and 30 Hz, providing both global workspace information and close-range views of bimanual interactions. The leader arms record operator commands and drive the corresponding follower arms. During model execution, the same leader-follower interface allows an operator to take over and issue corrective actions when the policy produces an erroneous action sequence. The bimanual configuration and synchronized multi-view sensing support real-world reinforcement learning experiments involving coordinated manipulation, contact-rich interactions, and long-horizon action sequences.

A.7 DoBot X-Trainer

Figure 10 shows the DoBot X-Trainer bimanual robot platform used in our experiments. The platform consists of two DoBot Nova2 six-degree-of-freedom robotic arms and three Intel RealSense D405 cameras. One camera is mounted above the workspace, while the other two are mounted on the wrists of the left and right arms, respectively. The overhead camera provides a global view of the workspace, and the wrist cameras capture local visual interactions between each end effector and the manipulated object. The platform is also equipped with a pair of Gello-style leader devices that independently control the two arms. During model inference, an operator can use the leader devices to intervene in either arm when the policy produces an erroneous action or requires correction, enabling the collection of demonstrations and the execution of corrective actions and recovery behaviors. This configuration provides a complementary hardware platform for evaluating the transfer of Efficient RLT across different robotic arms, camera placements, and teleoperation interfaces.

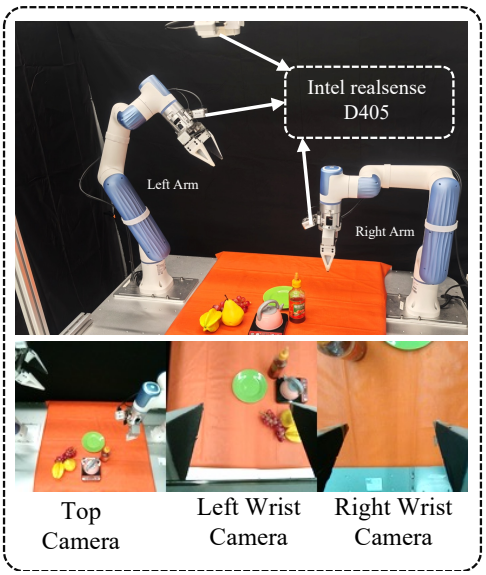


Figure 10: The DoBot X-Trainer platform used in our experiments. The platform consists of two DoBot Nova2 six-degree-of-freedom arms, three Intel RealSense D405 cameras, and a pair of leader devices for bimanual teleoperation and intervention.

A.8 G1-D MILESTONE COUNTS

Section **??** replaces the frozen $\pi_{0.5}$ base with EWAM. All three variants share that initialization, and Efficient-RLT adapts only the residual head. The task is *Pouring goji berries* (Fig. 6): grasp both cups, bring them together, pour the goji berries from one cup into the other, and place both cups back. A rollout sums the milestones in Table 4; completing none scores zero. Each variant is evaluated for 10 rollouts. Teleoperation uses a PICO 4 Ultra.

EWAM grasps the right cup in 9 of 10 rollouts and the left cup in 3, and brings the cups together in 3. Without rewind, left-cup grasps rise to 7 and right-cup grasps fall to 3, so the mean falls from 2.7 to 2.5. Full Efficient-RLT keeps both grasps (10 and 9) and brings the cups together in 8. Of the 2.2 points over EWAM, 1.4 come from the left-cup grasp and 0.5 from alignment; pouring occurs once. Placement is completed only in one no-rewind rollout, which placed both cups without pouring.

Table 3: Milestone completion on G1-D *Pouring goji berries*, out of 10 rollouts. The mean score averages the 10-point totals.

Milestone	Points	EWAM	w/o rewind	Efficient-RLT
Grasp the left cup	2	3	7	10
Grasp the right cup	2	9	3	9
Lift both and bring together	1	3	3	8
Pour	3	0	0	1
Place the left cup	1	0	1	0
Place the right cup	1	0	1	0
Mean score		2.7	2.5	4.9

A.9 TASK-SPECIFIC PROCESS SCORING

Table 4 lists the scoring milestones for the robot tasks in the main comparison and ablation studies. The process score is the sum of the points for completed milestones, with each milestone counted at most once and a maximum of 10 points per rollout.

Table 4: Task-specific criteria for the 10-point process score. Points are awarded for completing each listed milestone.

Task	Scoring milestone	Points
Franka		
Place Ring	Move to the side of the ring.	2
	Pick up the ring.	3
	Move the ring to the side of the rod.	2
	Insert the ring to the rod.	3
Insert Peg	Move the peg to the side of the corresponding hole.	5
	Insert the peg into the hole.	5
Stack Bowls	Move to the side of the first bowl.	1
	Pick up the first bowl.	2
	Successfully place the first bowl.	2
	Move to the side of the second bowl.	1
	Pick up the second bowl.	2
	Successfully place the second bowl.	2

Continued on next page

Table 4 (continued)

Task	Scoring milestone	Points
Place Fruits	Move to the side of the first fruit.	1
	Pick up the first fruit.	1
	put the first fruit to the plate.	1
	Move to the side of the second fruit.	1
	Pick up the second fruit.	1
	put the second fruit to the plate.	1
	Move to the side of the third fruit.	1
	Pick up the third fruit.	1
	put the third fruit to the plate.	2
CoBot		
Assemble Parts	Pick up a part with the right gripper.	2
	Insert the part into its corresponding hole with the right gripper.	3
	Pick up a part with the left gripper.	2
	Insert the part into its corresponding hole with the left gripper.	3
Store Block	Grasp the drawer handle with the left gripper.	3
	Pull the drawer open with the left gripper.	2
	Grasp the block with the right gripper.	1
	Place the block inside the drawer with the right gripper.	2
	Close the drawer with the left gripper.	2
Collect and Pour	Grasp the fruit with the left gripper.	2
	Place the fruit into the bowl with the left gripper.	3
	Grasp the bowl with the right gripper.	2
	Pour the fruit from the bowl into the designated basket.	3
Cook Vegetables	Grasp the vegetables with one gripper.	2
	Place the vegetables into the pot.	3
	Grasp the lid with the other gripper.	2
	Cover the pot with the lid.	3
DoBot		
Pour Water	Grasp and place the bowl at the target position.	3
	Grasp the pitcher.	3
	Pour water into the bowl.	2
	Place the pitcher down.	2
Fold Towel	Grasp the towel with the right gripper.	2.5
	Fold the towel with the right gripper.	2.5
	Grasp the towel with the left gripper.	2.5
	Fold the towel with the left gripper.	2.5
Tidy Table	Grasp an object with the right gripper.	2.5
	Place the object into the bowl with the right gripper.	2.5
	Grasp an object with the left gripper.	2.5
	Place the object into the bowl with the left gripper.	2.5
X-Square		
Insert USB	left arm moves the side of power plug.	1
	left arm picks up the power plug.	2
	right arm moves to the side of USB connector.	1
	right arm grasp the USB connector.	2
	right arm moves the side of power plug.	1
	Insert the usb connector to the power plug.	3

Continued on next page

Table 4 (continued)

Task	Scoring milestone	Points
G1-D		
Pouring goji berries	Grasp the left cup.	2
	Grasp the right cup.	2
	Lift both cups and bring them together.	1
	Pour the goji berries from one cup into the other.	3
	Place the left cup on the table.	1
	Place the right cup on the table.	1