# dVLA-RL: Reinforcement Learning over Denoising Trajectories for Discrete Diffusion Vision-Language-Action Models

**Anonymous authors**
Paper under double-blind review

## Abstract

Vision-Language-Action (VLA) models have established a powerful paradigm for generalist robotic manipulation by grounding control into the semantic reasoning of large Vision-Language Models (VLMs). Prevailing architectures typically model actions continuously via diffusion or flow processes, or discretely through either autoregressive generation or parallel decoding. Recently, Discrete Diffusion VLAs (dVLAs) have emerged as a distinct alternative, unifying vision, language, and action into a single discrete token space via masked generative modeling. While this paradigm successfully combines iterative action refinement with unified representations, its training has thus far been restricted to Supervised Fine-Tuning (SFT), leaving the potential of Reinforcement Learning (RL) for further policy refinement largely unexplored. A fundamental challenge in RL for dVLAs is that the marginal probability of the final action remains intractable. To address this challenge, we propose **dVLA-RL**, which shifts policy optimization from the intractable marginal likelihood of the final action to a tractable trajectory-level objective defined over the sampled denoising path. By factorizing this objective across denoising transitions, **dVLA-RL** naturally accommodates variable denoising horizons. Leveraging this flexibility, we introduce a unified step scheduling approach for complex multi-task learning, tailoring denoising steps to specific task complexities to improve the performance–efficiency trade-off. Extensive evaluations show that **dVLA-RL** achieves **99.7%** success on LIBERO and establishes strong VLA-based results on RoboTwin 2.0, improving over the SFT baseline by **30.6** percentage points while remaining competitive with strong World-Action Model baselines. On real-world tasks, **dVLA-RL** improves the aggregate score by **25.8** percentage points over the MM-ACT baseline, with performance comparable to the representative continuous-action VLA $\pi_{0.5}$.

## 1 Introduction

Grounding multimodal perception into precise physical actions remains a central challenge in Embodied AI. Vision-Language-Action (VLA) models have advanced robotic control by framing it as a sequence generation task (Black et al., 2025b;a; Kim et al., 2025b;a). However, scaling such policies through Supervised Fine-Tuning (SFT) alone is constrained by the scarcity and cost of high-quality robot trajectories, while policies trained only on fixed demonstrations can generalize poorly when test-time states depart from the training distribution (Li et al., 2026a; Liu et al., 2025). Online Reinforcement Learning (RL) provides a complementary post-training route: it collects trajectories through environment interaction and uses outcome rewards to optimize task success, allowing the policy to learn from states and behaviors not covered by the demonstration set (Li et al., 2026a; Liu et al., 2025). However, applying policy-gradient RL to VLA policies requires a tractable policy likelihood for the generated action, making the action-generation paradigm a crucial design factor.

Existing VLA action-generation paradigms employ distinct RL optimization strategies tailored to their underlying probability representations. Autoregressive and parallel decoding policies formulate action generation through token-wise or single-step predictions, allowing standard policy-gradient algorithms to directly utilize their exact analytical action likelihoods. Conversely, continuous diffusion and flow-matching policies focus on the expressive modeling of complex, multimodal



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.

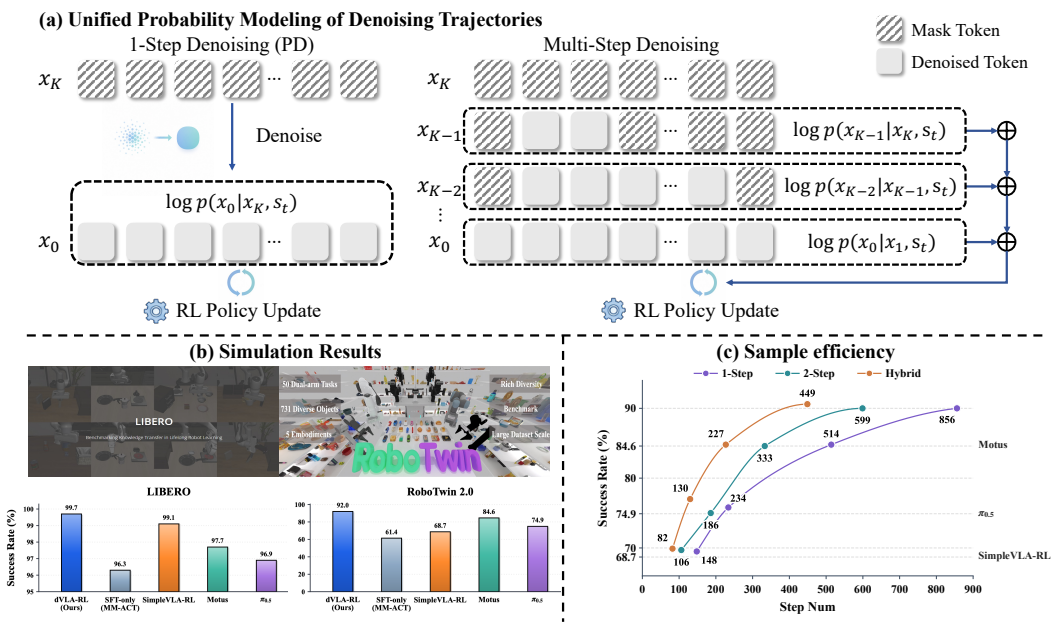



Figure 1: **Overview of dVLA-RL. (a)** We present a unified probability modeling approach for denoising trajectories, seamlessly integrating 1-step parallel decoding and multi-step denoising paradigms. **(b)** Extensive evaluations on the LIBERO and RoboTwin 2.0 benchmarks demonstrate that our RL-finetuned policy outperforms both SFT-only baselines and remains highly competitive among modern VLA architectures, achieving performance comparable to World-Action Models. **(c)** The Hybrid denoising strategy exhibits superior sample efficiency, achieving higher success rates with substantially fewer environment interaction steps.

action distributions. To make RL tractable in these continuous spaces, recent VLA methods introduce auxiliary stochastic transitions or convert deterministic flows into stochastic policies with computable likelihoods (Chen et al., 2026b).

Recently, Discrete Diffusion VLAs (dVLAs) have emerged as a promising new paradigm that integrates visual, linguistic, proprioceptive, and action modalities within a unified token-based framework, generating discrete action chunks through masked denoising (Liang et al., 2025). While recent works demonstrate strong Supervised Fine-Tuning (SFT) performance comparable to continuous counterparts (Liang et al., 2026; 2025; Chen et al., 2026a; Wen et al., 2025; Ye et al., 2025; Liu et al., 2026), Reinforcement Learning (RL) for native dVLAs remains largely underexplored. The primary bottleneck is that standard policy-gradient methods require the likelihood of the executed action, whereas a dVLA can reach the same final action through many intermediate denoising paths. Computing the exact marginal probability $\pi_\theta(a_t \mid s_t)$ therefore requires summing over all such paths. Simple last-step approximations, on the other hand, ignore the intermediate states that determine the rollout, preventing the direct application of standard RL objectives to dVLA policies.

In this work, we address this gap by formulating RL directly over the denoising path executed by the dVLA. Rather than marginalizing over all paths that lead to the final action, we construct a trajectory-level objective over the sampled denoising path by aggregating the log-likelihoods of newly decoded tokens across the sampled denoising path. This formulation accommodates variable denoising horizons, which we leverage through a task-adaptive strategy to improve the trade-off between policy performance and computational efficiency.

Built on these ideas, we present **dVLA-RL**, a method for likelihood-based policy optimization of native discrete diffusion VLA policies. Empirically, we evaluate **dVLA-RL** on LIBERO and RoboTwin 2.0. The results show that **dVLA-RL** consistently improves its SFT-only discrete diffusion VLA backbone, achieves state-of-the-art performance on LIBERO, and obtains strong VLA-based performance on RoboTwin 2.0 while remaining competitive with strong World-Action Model baselines. Further analyses show that the trajectory-level objective improves over a final-step-only

alternative in the tested settings, and that the hybrid denoising-step strategy improves efficiency while preserving the benefits of multi-step action refinement. On real-world tasks, **dVLA-RL** improves the aggregate score by **25.8** percentage points over MM-ACT* (Liang et al., 2026), with performance comparable to the representative continuous-action VLA $\pi_{0.5}$ (Black et al., 2025a).

In summary, our core contributions are threefold:

- We formulate a trajectory-level optimization objective for native discrete diffusion VLAs that accounts for token-generation decisions across the sampled denoising path, providing a tractable alternative to evaluating the intractable marginal likelihood of final actions.
- Building on this formulation, we introduce a task-adaptive denoising-step strategy. By assigning task-wise denoising horizons according to task complexity and initial policy performance, this strategy reduces computational cost on easier tasks while preserving iterative action refinement for harder tasks, improving both effectiveness and efficiency.
- We empirically demonstrate that reinforcement learning can substantially improve discrete diffusion VLA policies beyond their SFT initialization. **dVLA-RL** achieves a success rate of **99.7%** on LIBERO and improves over the MM-ACT* SFT baseline by **30.6** percentage points on RoboTwin 2.0 and **25.8** percentage points on real-world task, highlighting the potential of native discrete diffusion as a promising action-generation paradigm for RL-optimized VLA policies and the effectiveness of our method.

## 2 Related Work

### 2.1 Diffusion Language, Vision-Language, and Vision-Language-Action Models

Discrete diffusion language models have recently emerged as an alternative to autoregressive sequence modeling. Instead of generating tokens in a fixed left-to-right order, they construct discrete sequences through masked denoising, enabling parallel prediction, bidirectional conditioning, and iterative refinement (Austin et al., 2021; Hoogeboom et al., 2021; Sahoo et al., 2024; Ye et al., 2025; Nie et al., 2026). This generation paradigm has also been extended to vision-language modeling, where diffusion-based VLMs unify multimodal understanding and generation by denoising visual and textual tokens under shared multimodal contexts (Yang et al., 2026; You et al., 2026). Compared with autoregressive VLMs, these models provide a more flexible mechanism for order-free multimodal generation and naturally support parallel refinement over multiple token types. Recent discrete diffusion VLA policies further extend this paradigm from language and vision-language generation to robotic action generation (Liang et al., 2026; 2025; Chen et al., 2026a; Wen et al., 2025; Ye et al., 2025; Liu et al., 2026), achieving performance compatible with state-of-the-art VLAs with continuous action modeling (Kim et al., 2025a; Black et al., 2025a).

However, existing discrete diffusion VLA studies are still largely centered on supervised fine-tuning with demonstration data, leaving reinforcement learning post-training underexplored.

### 2.2 Reinforcement Learning for Discrete Diffusion Models

Reinforcement learning for discrete diffusion is challenging because a generated sequence may arise through many denoising paths, making its marginal likelihood intractable. Existing methods therefore optimize endpoint-conditioned denoising surrogates constructed from corrupted final samples (Zhao et al., 2025; Yang et al., 2026; Gong et al., 2026), reweight timestep- and token-level gradients to improve stability (Zhang et al., 2026), or estimate step-wise values for intermediate decoding states (Wang et al., 2026). These approaches avoid evaluating the probability of the actual inference path and are primarily designed for language or image generation.

Embodied dVLAs present a distinct setting: their denoising chains are typically short (Liang et al., 2026), and intermediate action-token sequences serve only as internal refinement states rather than environment actions. These properties motivate trajectory-level optimization of the executed denoising process.

### 2.3 Reinforcement Learning for Vision-Language-Action Policies

More broadly, RL has been introduced into VLA training as a post-training mechanism that directly optimizes task outcomes beyond supervised imitation. Recent work applies PPO- or GRPO-style post-training to autoregressive, tokenized, and regression-based VLAs (Lu et al., 2025; Tan et al., 2025; Chen et al., 2025b; Li et al., 2026a). Scalable systems further support efficient VLA rollout and training (Yu et al., 2025; Zang et al., 2026). For continuous-action VLAs, complementary approaches either directly optimize flow-based generators or train a lightweight actor–critic interface on top of a pretrained VLA (Chen et al., 2026b; Xu et al., 2026). However, to the best of our knowledge, **dVLA-RL** is the first work to apply online RL directly to native discrete diffusion VLA policies.

## 3 Preliminaries

### 3.1 Discrete Diffusion Action Generation

At an environment state $s$, comprising visual observations, a language instruction, and robot proprioception, a dVLA represents the executable action as a discrete sequence $x_0$. Generation starts from a fully masked sequence $x_K$ and proceeds through $K$ denoising steps. At step $k$, the network predicts per-token categorical distributions $p_\theta(\cdot \mid x_k, s)$, and a scheduler selects a subset of token indices $M_k$ to decode, together inducing the transition $P_\theta(x_{k-1} \mid x_k, s)$. We denote the complete internal generation path by $\tau = (x_K, x_{K-1}, \ldots, x_0)$.

### 3.2 PPO and the Marginal-Likelihood Problem

At environment timestep $t$, the policy observes $s_t$, generates the action $a_t = x_0$ through its internal denoising chain, and receives an environment reward. Proximal Policy Optimization uses the clipped surrogate objective

$$\begin{aligned} L^{\mathrm{CLIP}}(\theta) &= \mathbb{E}_t[\min(\rho_t(\theta)A_t, \mathrm{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\, A_t)]\,, \\ \rho_t(\theta) &= \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t \mid s_t)}, \end{aligned} \tag{1}$$

where $A_t$ is the advantage estimate and $\rho_t(\theta)$ is the importance-sampling ratio.

For a dVLA, however, evaluating the exact probability of the executed action requires marginalizing over every intermediate denoising path that produces $a_t = x_0$:

$$\pi_\theta(a_t \mid s_t) = \sum_{x_1,\ldots,x_{K-1}} \prod_{k=1}^{K} P_\theta(x_{k-1} \mid x_k, s_t), \qquad x_K = \texttt{[MASK]},\ x_0 = a_t. \tag{2}$$

The number of valid paths grows combinatorially, whereas any single-step distribution $p_\theta(\cdot \mid x_k, s_t)$ describes only a local transition. Consequently, using a local transition probability as the policy likelihood omits the other denoising transitions that jointly produce the executed action and does not faithfully represent the full generation process. Section 4 addresses this mismatch by constructing a tractable trajectory-level objective over the particular denoising path sampled during rollout.

## 4 Method

### 4.1 Trajectory-Level Likelihood of Denoising Paths

To avoid evaluating the intractable marginal action probability $\pi_\theta(a_t \mid s_t)$ discussed in Section 3.2, we optimize the token-generation likelihood along the denoising path observed during rollout. Let $\tau = (x_K, \ldots, x_0)$ denote the sequence of denoising states, and let $\mathcal{M} = (M_K, \ldots, M_1)$ record the positions that become newly decoded at each step. Under this fixed schedule, the likelihood of the committed token-generation decisions factorizes as

$$\pi_\theta^{\mathrm{tok}}(\tau \mid s_t, \mathcal{M}) := \prod_{k=1}^{K} \prod_{i \in M_k} p_\theta(x_{k-1}^i \mid x_k, s_t). \tag{3}$$

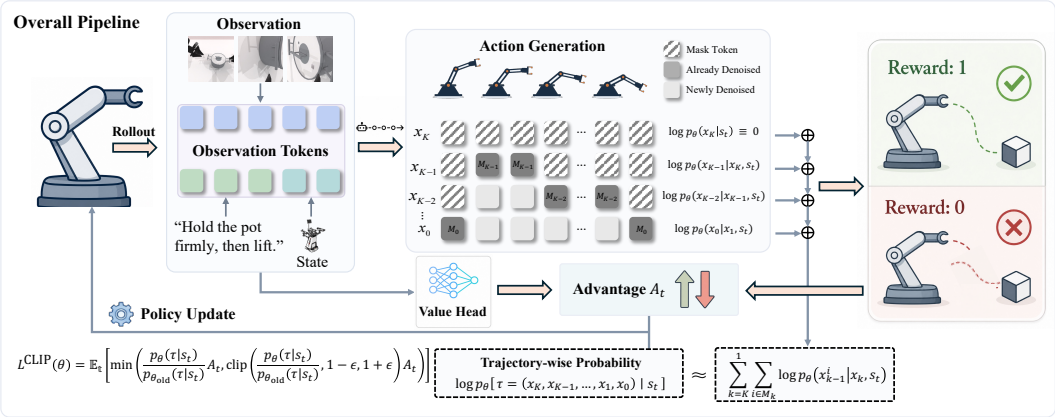



Figure 2: **The overall pipeline of dVLA-RL.** (1) **Rollout & Action Generation:** During interaction, the agent conditions on multimodal observations $s_t$ to generate actions via a $K$-step iterative masked denoising process, executing them to collect trajectory rewards. (2) **Trajectory-Level Policy Optimization:** Conditioned on the scheduler decisions recorded during rollout, we aggregate the log-likelihoods of newly decoded tokens across the denoising path. Combined with advantages $A_t$ estimated by the value head, this trajectory-level token likelihood is optimized with the PPO clipped objective.

Previously decoded tokens are copied to the next state, while positions that remain masked do not yet contribute a committed token value. Therefore, each action token is counted once, at the step where it first transitions from `[MASK]` to its decoded value. The corresponding trajectory-level log-likelihood is

$$\ell_\theta(\tau \mid s_t, \mathcal{M}) = \sum_{k=1}^{K} \sum_{i \in M_k} \log p_\theta(x_{k-1}^i \mid x_k, s_t). \tag{4}$$

### 4.2 Policy Optimization with a Frozen Scheduler

At each policy iteration, the behavior parameters $\bar{\theta} = \theta_{\text{old}}$ are first used to collect rollouts. Our deterministic confidence-based Top-K scheduler determines $\mathcal{M}$, and we record the newly decoded positions and unknown-position masks needed to reconstruct every intermediate denoising state. During the subsequent policy update, these records are replayed and the scheduler is held fixed. Thus, the current and behavior token generators are compared under the same denoising schedule, giving the likelihood ratio

$$\widehat{\rho}_t(\theta) = \frac{\pi_\theta^{\text{tok}}(\tau_t \mid s_t, \mathcal{M}_t)}{\pi_{\bar{\theta}}^{\text{tok}}(\tau_t \mid s_t, \mathcal{M}_t)} = \exp(\ell_\theta(\tau_t \mid s_t, \mathcal{M}_t) - \ell_{\bar{\theta}}(\tau_t \mid s_t, \mathcal{M}_t)). \tag{5}$$

Formally, one policy update defines an auxiliary policy with the recorded schedule as a fixed operator,

$$\widetilde{\pi}_{\theta|\mathcal{M}_t}(\tau_t \mid s_t) = q_t^{\text{fix}}(\mathcal{M}_t)\, \pi_\theta^{\text{tok}}(\tau_t \mid s_t, \mathcal{M}_t), \tag{6}$$

where $q_t^{\text{fix}}$ is a point mass on the recorded schedule $\mathcal{M}_t$ and is shared by the current and behavior policies. It therefore cancels exactly in the auxiliary likelihood ratio. Accordingly, $\pi_\theta^{\text{tok}}$ represents the likelihood of committed token decisions conditioned on the fixed schedule. The resulting $\widehat{\rho}_t$ is therefore a likelihood ratio conditioned on the recorded schedule, which we use in the policy objective below in place of the intractable marginal action-probability ratio.

We substitute $\widehat{\rho}_t(\theta)$ for the intractable action-level ratio in the clipped policy objective:

$$L^{\textbf{dVLA-RL}}(\theta) = \mathbb{E}_t[\min(\widehat{\rho}_t(\theta)A_t, \text{clip}(\widehat{\rho}_t(\theta), 1-\epsilon, 1+\epsilon)\, A_t)]. \tag{7}$$

Here $A_t$ is the chunk-level advantage shared by all token decisions that generate the action at environment step $t$. This defines an alternating surrogate optimization step: token generation is updated under the frozen recorded schedule, without differentiating through the discrete Top-K selections. Before the next rollout phase, the behavior parameters are refreshed from the updated policy, so the scheduler is recomputed from the new predictions between iterations while remaining fixed within each policy update.

# 5 EXPERIMENTS

Our experiments evaluate **dVLA-RL** along four dimensions. First, we compare it with its matched SFT-only backbone and representative VLA and World Action Model baselines on LIBERO and RoboTwin 2.0. Second, we ablate the proposed trajectory-level objective against a final-step-only alternative. Third, we analyze fixed and task-adaptive denoising horizons in terms of task performance, sample efficiency, and computational cost. Finally, we evaluate its real-world manipulation performance on the ALOHA platform.

## 5.1 EXPERIMENTAL SETUP

**Benchmarks and initialization.** We evaluate on LIBERO (Liu et al., 2023), whose Spatial, Object, Goal, and Long suites test spatial reasoning, object-centric manipulation, goal variation, and long-horizon composition, and RoboTwin 2.0 (Chen et al., 2025a), which adds bimanual coordination, diverse objects, and domain-randomized scenes. We instantiate **dVLA-RL** on MM-ACT (Liang et al., 2026), a discrete diffusion VLA that represents action chunks as tokens and generates them through masked denoising. Following the setting of SimpleVLA-RL (Li et al., 2026a), we use 500 SFT demonstrations per LIBERO suite and 1,000 per RoboTwin task.

Our matched SFT-only control, MM-ACT$^*$, shares the architecture and initialization of **dVLA-RL** but receives no online RL, thereby isolating the effect of post-training. Relative to the published MM-ACT recipe, it uses action-only SFT, action chunks of length 16, and joint multi-task training. Appendix B provides the complete SFT setup. We additionally report representative VLA and World Action Model (WAM) results to contextualize absolute performance, while the matched MM-ACT$^*$ comparison remains the primary controlled measure of the effect of RL.

**Simulation RL and evaluation.** Starting from the SFT checkpoint, we run online PPO (Schulman et al., 2017) in RLinf (Yu et al., 2025) with a sparse reward of one for success and zero otherwise, and report success rate (SR). SFT evaluation uses 50 trials per LIBERO subtask (2,000 total) and 100 trials per RoboTwin task under four-step denoising. For **dVLA-RL**, the tables report the best online SR observed during training: 512 rollout episodes per LIBERO task, and 64 per RoboTwin task drawn from a pool of 1,000 training scenarios. For RoboTwin, we select the checkpoint with the highest eight-task average and report its corresponding per-task SRs. Appendices C and D provide complete hyperparameters and sampling protocols.

## 5.2 DOES dVLA-RL IMPROVE DISCRETE DIFFUSION VLA POLICIES?

**LIBERO.** Table 1 shows that **dVLA-RL** exceeds 99% SR on every suite and reaches a **99.7%** average. This improves the matched MM-ACT$^*$ backbone from $88.1\%$ by 11.6 points, demonstrating that sparse-reward online RL supplies task-level supervision beyond behavior cloning. The gains hold across spatial reasoning, object manipulation, goal variation, and long-horizon composition rather than being concentrated in one suite. The resulting average also exceeds all listed VLA and WAM baselines, showing that online optimization of the native denoising policy can retain strong multi-task performance.

**RoboTwin 2.0.** Table 2 shows that **dVLA-RL** raises average SR from $61.4\%$ to $92.0\%$ ($+30.6$ points) and improves all eight tasks. These tasks span grasping, placing, object transfer, and tool–object interaction. The largest gains occur on Handover Mic ($+47.9$), Move Can Pot ($+47.5$), and Lift Pot ($+43.1$), indicating that the benefit extends to bimanual coordination and longer closed-loop execution rather than only easier single-arm behaviors. **dVLA-RL** is the strongest listed VLA result on this visually challenging, domain-randomized benchmark.

Table 1: LIBERO success rate (%) by suite and four-suite average.

| Method | Spatial | Object | Goal | Long | Avg. |
|---|---|---|---|---|---|
| **# VLA-Based** | | | | | |
| OpenVLA (Kim et al., 2025b) | 84.7 | 88.4 | 79.2 | 53.7 | 76.5 |
| $\pi_0$ (Black et al., 2025b) | 96.8 | 98.8 | 95.8 | 85.2 | 94.2 |
| UniVLA (Bu et al., 2025) | 96.5 | 96.8 | 95.6 | 92.0 | 95.2 |
| $\pi_{0.5}$ (Black et al., 2025a) | 98.8 | 98.2 | 98.0 | 92.4 | 96.9 |
| OpenVLA-OFT (Kim et al., 2025a) | 97.6 | 98.4 | 97.9 | 94.5 | 97.1 |
| SimpleVLA-RL (Li et al., 2026a) | 99.4 | 99.1 | 99.2 | 98.5 | 99.1 |
| MM-ACT (Liang et al., 2026) | 97.8 | 99.4 | 94.8 | 93.0 | 96.3 |
| MM-ACT* | 91.2 | 82.6 | 90.0 | 88.7 | 88.1 |
| **dVLA-RL (Ours)** | **99.8** | **100.0** | **99.6** | **99.2** | **99.7** |
| **# WAM-Based** | | | | | |
| Motus (Bi et al., 2026) | 96.8 | 99.8 | 96.6 | 97.6 | 97.7 |
| Cosmos Policy (Kim et al., 2026) | 98.1 | 100.0 | 98.2 | 97.6 | 98.5 |
| LingBot-VA (Li et al., 2026b) | 98.5 | 99.6 | 97.2 | 98.5 | 98.5 |
| Fast-WAM (Yuan et al., 2026) | 98.2 | 100.0 | 97.0 | 95.2 | 97.6 |

Table 2: RoboTwin 2.0 success rate (%) on eight tasks. BBH: Beat Block Hammer; PPS: Place Phone Stand; PDB: Pick Dual Bottles; LP: Lift Pot; MCP: Move Can Pot; PAL: Place A2B Left; PEC: Place Empty Cup; HM: Handover Mic. Gray rows are cross-protocol WAM references.

| Method | BBH | PPS | PDB | LP | MCP | PAL | PEC | HM | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| **# VLA-Based** | | | | | | | | | |
| $\pi_0$ | 59.0 | 22.0 | 50.0 | 51.0 | 41.0 | 38.0 | 60.0 | 96.0 | 52.1 |
| RDT | 22.0 | 13.0 | 18.0 | 45.0 | 33.0 | 21.0 | 42.0 | 95.0 | 36.1 |
| OpenVLA-OFT | 28.1 | 17.1 | 29.7 | 10.1 | 28.1 | 37.5 | 77.3 | 45.3 | 34.2 |
| SimpleVLA-RL | 87.5 | 39.6 | 68.3 | 64.1 | 61.2 | 45.3 | 94.2 | 89.2 | 68.7 |
| $\pi_{0.5}$ | 88.0 | 73.0 | 98.0 | 88.0 | 72.0 | 74.0 | 74.0 | 32.0 | 74.9 |
| MM-ACT* | 85.0 | 62.0 | 71.0 | 46.0 | 40.0 | 57.0 | 81.0 | 49.0 | 61.4 |
| **dVLA-RL (Ours)** | **95.3** | **96.9** | **95.3** | **89.1** | **87.5** | **79.7** | **95.3** | **96.9** | **92.0** |
| **# WAM-Based** | | | | | | | | | |
| Motus | 88.0 | 86.0 | 90.0 | 99.0 | 74.0 | 79.0 | 98.0 | 63.0 | 84.6 |
| GigaWorld | 86.0 | 72.0 | 86.0 | 98.0 | 78.0 | 88.0 | 90.0 | 72.0 | 83.8 |
| LingBot-VA | 98.0 | 97.0 | 99.0 | 99.0 | 97.0 | 93.0 | 100.0 | 96.0 | 97.4 |
| Fast-WAM | 97.0 | 99.0 | 96.0 | 100.0 | 88.0 | 93.0 | 100.0 | 100.0 | 96.6 |

The WAM rows provide absolute reference points (Bi et al., 2026; Ye et al., 2026; Li et al., 2026b; Yuan et al., 2026), but are not controlled comparisons because they follow different RoboTwin training protocols. We therefore treat the gain over the matched SFT backbone as the primary comparison.

### 5.3 Does Trajectory-Level Optimization Improve over a Final-Step Objective?

A dVLA produces an action through several conditional denoising transitions, so a likelihood computed only at the final step does not represent the path executed during rollout. Our trajectory-level objective instead accounts for the sequence of denoising transitions that generates the executable action.

We isolate this modeling choice on LIBERO-Spatial and LIBERO-Object by comparing our trajectory-level objective with a final-step-only proxy. The backbone, rollout budget, sparse reward, and all settings other than the likelihood score are held fixed, and both variants are tested with two- and four-step decoding. Across both tasks and horizons, the trajectory-level objective improves more consistently (Appendix Figure 5). On Spatial, the proxy becomes unstable with two steps and plateaus lower with four; on Object, it degrades later in training, whereas trajectory-level optimization remains near saturation for both horizons. These results show the empirical benefit of accounting for the complete denoising trajectory rather than only its final step in the tested settings. Appendix C.1 provides the complete curves and analysis.

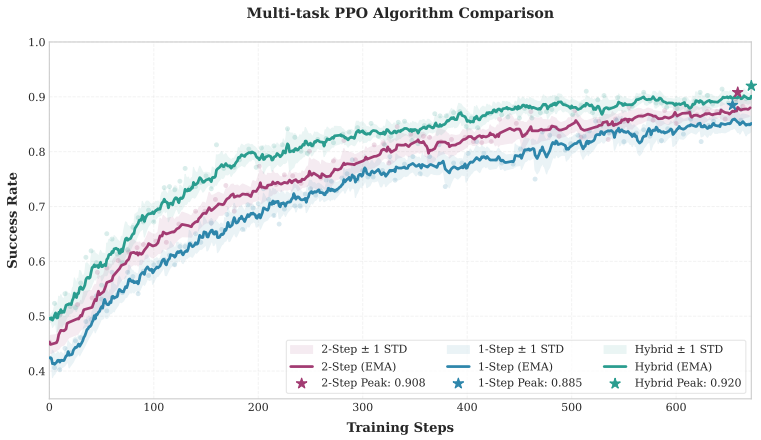



Figure 3: RoboTwin 2.0 training curves for fixed one-step, fixed two-step, and task-adaptive Hybrid denoising. Curves show EMA SR with $\pm 1$ standard deviation; stars mark peak SR. Hybrid uses one, two, or four steps across tasks.

### 5.4 How Do Denoising Steps Affect Optimization and Efficiency?

Different horizons produce denoising trajectories of different lengths. One-step decoding collapses generation into a single parallel transition; two steps add an intermediate state; and four steps permit finer iterative correction. Better-refined SFT actions can raise the initial success rate, which is particularly valuable under sparse rewards because PPO then encounters informative positive trajectories earlier. The trajectory-level formulation accommodates every horizon by evaluating its corresponding executed denoising path without changing the form of the policy objective.

**dVLA-RL** exploits this flexibility through a task-adaptive denoising strategy, termed Hybrid, which assigns one, two, or four steps to each task. We evaluate the SFT checkpoint at different horizons and select the smallest setting whose SR remains comparable to longer-step decoding. This strategy retains iterative refinement where it materially improves the initial policy while avoiding unnecessary transitions on tasks already solved well with fewer steps; Appendix Table 7 gives the assignments.

Figure 3 confirms the link between initialization and sparse-reward optimization. Fixed two-step decoding starts stronger and learns faster than one-step decoding, while Hybrid maintains the highest SR through most of training and reaches the high-success regime earlier. Under the same training budget, their peak SRs are $0.908$, $0.885$, and $0.920$ for fixed two-step, fixed one-step, and Hybrid, respectively. The result shows that minimizing the cost of each action generation is not identical to maximizing sample efficiency: selectively retaining refinement can expose PPO to more useful trajectories. Overall, the horizon affects RL through two coupled mechanisms: it controls iterative refinement before and during RL, and it defines the trajectory structure over which policy probabilities are evaluated and optimized. Appendix D.1 provides the full optimization analysis and per-task curves.

### 5.5 Does Hybrid Scheduling Improve Computational Efficiency?

Fixed multi-step decoding improves action refinement but increases both rollout and deployment cost almost proportionally to the number of transitions. Hybrid avoids paying the four-step cost on every task while preserving extra refinement where it improves exploration. Training time denotes wall-clock RL cost, whereas inference delay and FLOPs are measured per generated action chunk. Relative to fixed four-step decoding, Hybrid reduces training time from $845.95$ s to $607.89$ s ($28.1\%$), and per-chunk inference delay and FLOPs from $728.92$ ms and $70{,}191.28$ GFLOPs to $387.24$ ms and $37{,}289.12$ GFLOPs (both $46.9\%$). Fixed one-step decoding is cheaper, but its lower peak SR ($0.885$ versus $0.920$) reveals a weaker performance–efficiency trade-off. Hybrid therefore improves computational efficiency without giving up the optimization benefit of task-dependent refinement. Appendix Table 8 reports every setting, and Appendix D.2 details the measurements.

Table 3: Real-world performance (%) on ALOHA. PEC: Place Empty Cup; LP: Lift Pot; HM: Handover Mic; OT: Organize Table. OT reports stage success rates for PPC, lifting and moving the pot (LP), and placing the cup (PEC). OT columns report individual stage success rates, not full-sequence success. Avg. is the arithmetic mean of PEC, LP, HM, and the average success rate across the three OT stages. Column-best results are bold.

| Method | PEC | LP | HM | OT | | | Avg. |
|---|---|---|---|---|---|---|---|
| | | | | PPC | LP | PEC | |
| MM-ACT* | 60.0 | 50.0 | 55.0 | 80.0 | 50.0 | 40.0 | 55.4 |
| $\pi_{0.5}$ | **90.0** | 70.0 | 75.0 | 90.0 | 70.0 | 65.0 | 77.5 |
| **dVLA-RL (Ours)** | 85.0 | **75.0** | **80.0** | **95.0** | **85.0** | **75.0** | **81.3** |

### 5.6 Real World Experiments

We evaluate **dVLA-RL** on the dual-arm ALOHA platform (Zhao et al., 2023) using three short-horizon tasks, *Place Empty Cup*, *Lift Pot*, and *Handover Mic*, adapted from RoboTwin, and one long-horizon task, *Organize Table*. The latter comprises three sequential stages: transferring a carrot from a plate into a pot (Pick and Place Carrot, PPC), lifting and moving the pot with both arms to reveal a coaster underneath, and placing a cup onto the coaster. We collect 100 demonstrations per short-horizon task and 150 for Organize Table, totaling 450 demonstrations. For real-world adaptation, we fine-tune the **dVLA-RL** checkpoint trained on RoboTwin and its SFT-only MM-ACT* counterpart using these demonstrations; no RL is performed on physical robots. We evaluate each of the four tasks over 20 trials. Further details are provided in Appendix E.

Table 3 shows that **dVLA-RL** achieves an aggregate score of **81.3%**, improving over MM-ACT* by 25.8 percentage points. Average success on the three short-horizon tasks increases from $55.0\%$ to $80.0\%$, with a 25-point gain on each task. On Organize Table, mean stage success rises from $56.7\%$ to $85.0\%$, with the largest gains in pot moving and cup placement ($+35$ points each). Overall, **dVLA-RL** achieves performance comparable to $\pi_{0.5}$, a representative continuous-action VLA.

Notably, $\pi_{0.5}$ benefits from extensive real-robot pretraining, whereas MM-ACT* and **dVLA-RL** receive no such pretraining before task-specific adaptation. During RL on RoboTwin, **dVLA-RL** optimizes task success on self-generated trajectories in randomized environments, potentially learning action patterns that tolerate execution-induced deviations and environmental variation. These benefits may persist through real-world adaptation, helping accommodate diverse object and gripper configurations at stage transitions, compounded by sensing and execution noise.

## 6 Conclusion

In this work, we presented **dVLA-RL**, a method for likelihood-based policy optimization of discrete diffusion Vision-Language-Action policies. Rather than approximating the intractable marginal likelihood of the final action, **dVLA-RL** constructs a tractable trajectory-level surrogate by aggregating the likelihoods of newly decoded tokens along the sampled denoising path.

Empirically, **dVLA-RL** consistently improves its SFT-only discrete diffusion VLA backbone and achieves strong performance on both LIBERO and RoboTwin 2.0. On real-world ALOHA tasks, **dVLA-RL** improves the aggregate score by 25.8 percentage points over MM-ACT*, with performance comparable to the representative continuous-action VLA $\pi_{0.5}$. Beyond overall success rates, our analyses show that the trajectory-level objective improves over a final-step-only alternative in the tested settings, while multi-step and mixed-step denoising objectives lead to more favorable RL optimization. Furthermore, the proposed Hybrid strategy leverages the flexibility of the trajectory-level formulation to balance task performance and computational efficiency by adapting the denoising steps used during training and inference. These results demonstrate that native discrete diffusion policies, when equipped with trajectory-level RL optimization, provide a promising foundation for unified and efficient robotic action generation.

### AI USE STATEMENT

Generative AI tools were used to assist with translation, language editing, and improving the clarity, organization, and presentation of the manuscript. All AI-assisted text was reviewed and revised by the authors, who take responsibility for the final content.

### ETHICS STATEMENT

This work evaluates robotic manipulation policies in simulation and controlled physical-robot experiments and does not involve human subjects or personal data. Deployment on physical robots should include task-specific safety validation and appropriate human oversight.

### REPRODUCIBILITY STATEMENT

We support reproducibility by providing the algorithm in Appendix A, the SFT setup in Appendix B, the online RL and evaluation protocol in Section 5.1, and benchmark-specific training details in Appendices C and D.

## REFERENCES

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. *CoRR*, abs/2107.03006, 2021. URL https://arxiv.org/abs/2107.03006.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35101–35113, 2026.

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

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert 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, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. $\pi_0$: A Vision-Language-Action Flow Model for General Robot Control. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, 2025b. doi: 10.15607/RSS.2025.XXI.010.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Learning to Act Anywhere with Task-centric Latent Actions. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.014.

Jiayi Chen, Wenxuan Song, Pengxiang Ding, Ziyang Zhou, Han Zhao, Feilong Tang, Donglin Wang, and Haoang Li. Unified diffusion VLA: Vision-language-action model via joint discrete denosing diffusion process. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=UvQOcw2oCD.

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. $\pi_{\text{RL}}$: Online

rl fine-tuning for flow-based vision-language-action models, 2026b. URL https://arxiv.org/abs/2510.25889.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation, 2025a. URL https://arxiv.org/abs/2506.18088.

Zengjue Chen, Runliang Niu, He Kong, Qi Wang, Qianli Xing, and Zipei Fan. Tgrpo :fine-tuning vision-language-action model via trajectory-wise group relative policy optimization, 2025b. URL https://arxiv.org/abs/2506.08440.

Shansan Gong, Ruixiang ZHANG, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. Diffucoder: Understanding and improving masked diffusion models for code generation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=58NA3unZj5.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in neural information processing systems*, 34:12454–12465, 2021.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025a. doi: 10.15607/RSS.2025.XXI.017.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R 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. 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. 2679–2713. PMLR, 2025b. URL https://proceedings.mlr.press/v270/kim25c.html.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=wPEIStHxYH.

Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Yang Zhaohui, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, Dehui Wang, Dingxiang Luo, Yuchen Fan, Youbang Sun, Jia Zeng, Jiangmiao Pang, Shanghang Zhang, Yu Wang, Yao Mu, Bowen Zhou, and Ning Ding. SimpleVLA-RL: Scaling VLA training via reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=TQhSodCM4r.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal world modeling for robot control, 2026b. URL https://arxiv.org/abs/2601.21998.

Haotian Liang, Xinyi Chen, Bin Wang, Mingkang Chen, Yitian Liu, Yuhao Zhang, Zanxin Chen, Tianshuo Yang, Yilun Chen, Jiangmiao Pang, Dong Liu, Xiaokang Yang, Yao Mu, Wenqi Shao, and Ping Luo. Mm-act: Learn from multimodal parallel generation to act. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 35080–35090, June 2026.

Zhixuan Liang, Yizhuo Li, Tianshuo Yang, Chengyue Wu, Sitong Mao, Liuao Pei, Xiaokang Yang, Jiangmiao Pang, Yao Mu, and Ping Luo. Discrete diffusion vla: Bringing discrete diffusion to action decoding in vision-language-action policies, 2025. URL https://arxiv.org/abs/2508.20072.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, qiang liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=xzEtNSuDJk.

Jijia Liu, Feng Gao, Bingwen Wei, Xinlei Chen, Qingmin Liao, YI WU, Chao Yu, and Yu Wang. What can rl bring to vla generalization? an empirical study. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 97121–97151. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/8c0fabe372177d2aded596be2d3b4544-Paper-Conference.pdf.

Yang Liu, Pengxiang Ding, Tengyue Jiang, Xudong Wang, Wenxuan Song, Minghui Lin, Han Zhao, Hongyin Zhang, Zifeng Zhuang, Wei Zhao, et al. Mmada-vla: Large diffusion vision-language-action model with unified multi-modal instruction and generation. *arXiv preprint arXiv:2603.25406*, 2026.

Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. Vla-rl: Towards masterful and general robotic manipulation with scalable reinforcement learning, 2025. URL https://arxiv.org/abs/2505.18719.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *Advances in Neural Information Processing Systems*, 38:50608–50646, 2026.

Subham S Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

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

Shuhan Tan, Kairan Dou, Yue Zhao, and Philipp Krähenbühl. Interactive post-training for vision-language-action models, 2025. URL https://arxiv.org/abs/2505.17016.

Yinjie Wang, Ling Yang, Bowen Li, Ye Tian, Ke Shen, and Mengdi Wang. Revolutionizing reinforcement learning framework for diffusion large language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=KNAyc9DMe3.

Junjie Wen, Minjie Zhu, Jiaming Liu, Zhiyuan Liu, Yicun Yang, Linfeng Zhang, Shanghang Zhang, Yichen Zhu, and Yi Xu. dvla: Diffusion vision-language-action model with multimodal chain-of-thought. *CoRR*, abs/2509.25681, September 2025. URL https://doi.org/10.48550/arXiv.2509.25681.

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, 2026. URL https://arxiv.org/abs/2604.23073.

Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Mmada: Multimodal large diffusion language models. *Advances in Neural Information Processing Systems*, 38:138867–138907, 2026.

Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Hengtao Li, Jie Li, Jindi Lv, Jingyu Liu, Min Cao, Peng Li, Qiuping Deng, Wenjun Mei, Xiaofeng Wang, Xinze Chen, Xinyu Zhou, Yang Wang, Yifan Chang, Yifan Li, Yukun Zhou, Yun Ye, Zhichao Liu, and Zheng Zhu. Gigaworld-policy: An efficient action-centered world–action model, 2026. URL https://arxiv.org/abs/2603.17240.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025.

Zebin You, Shen Nie, Xiaolu Zhang, JUN ZHOU, Zhiwu Lu, Ji-Rong Wen, and Chongxuan Li. Llada-v: Large language diffusion models with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10093–10105, 2026.

Chao Yu, Yuanqing Wang, Zhen Guo, Hao Lin, Si Xu, Hongzhi Zang, Quanlu Zhang, Yongji Wu, Chunyang Zhu, Junhao Hu, Zixiao Huang, Mingjie Wei, Yuqing Xie, Ke Yang, Bo Dai, Zhexuan Xu, Jiakun Du, Xiangyuan Wang, Xu Fu, Letong Shi, Zhihao Liu, Kang Chen, Weilin Liu, Gang Liu, Boxun Li, Jianlei Yang, Zhi Yang, Guohao Dai, and Yu Wang. Rlinf: Flexible and efficient large-scale reinforcement learning via macro-to-micro flow transformation, 2025. URL https://arxiv.org/abs/2509.15965.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination?, 2026. URL https://arxiv.org/abs/2603.16666.

Hongzhi Zang, Mingjie Wei, Si Xu, Yongji Wu, Zhen Guo, Yuanqing Wang, Hao Lin, Peihong Wang, Liangzhi Shi, Yuqing Xie, Zhexuan Xu, Zhihao Liu, Kang Chen, Wenhao Tang, Quanlu Zhang, Weinan Zhang, Chao Yu, and Yu Wang. Rlinf-vla: A unified and efficient framework for reinforcement learning of vision-language-action models, 2026. URL https://arxiv.org/abs/2510.06710.

Mu Zhang, Tianren Ma, Yunfan Liu, Kun Hu, and Qixiang Ye. Rebrl: Reinforcing discrete visual diffusion models with rebalanced timestep credits. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 43135–43144, June 2026.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 56729–56762. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/52190a0362148d179f1cbd9080956872-Paper-Conference.pdf.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. URL https://arxiv.org/abs/2304.13705.

# A PSEUDOCODE OF DVLA-RL

**Algorithm 1** Unified Discrete VLA Reinforcement Learning (**dVLA-RL**)

**Input:** Pre-trained discrete diffusion VLA model $\pi_\theta$, Value network $V_\phi$, Environment $\mathcal{E}$
**Hyperparameters:** Denoising steps $K$, PPO clip ratio $\epsilon$, PPO epochs $E$, batch size $B$
**Initialize:** Policy parameters $\theta \leftarrow \theta_0$, Value parameters $\phi \leftarrow \phi_0$
**for** iteration $= 1, 2, \ldots$ **do**
  Set and freeze behavior parameters $\bar{\theta} \leftarrow \theta$
  # Phase 1: Environment Rollout & Trajectory Sampling
  Initialize experience buffer $\mathcal{D} = \emptyset$
  **for** $t = 1 \ldots T_{\text{env}}$ **do**
    Observe multi-modal state $s_t$ (images, text instructions, proprioception)
    Initialize fully masked action sequence $x_K = \texttt{[MASK]}^N$
    **for** $k = K$ **down to** $1$ **do**
      Record the unknown positions $H_k = \{i : x_k^i = \texttt{[MASK]}\}$
      Compute token distributions $p_{\bar{\theta}}(\cdot|x_k, s_t)$
      Sample provisional tokens $\widetilde{x}^i \sim p_{\bar{\theta}}(\cdot \mid x_k, s_t)$ for $i \in H_k$
      Compute confidence $c_i = p_{\bar{\theta}}(\widetilde{x}^i \mid x_k, s_t)$ for $i \in H_k$
      Select committed positions $M_k \subseteq H_k$ using the deterministic confidence/Top-K scheduler
      Set $x_{k-1}^i \leftarrow \widetilde{x}^i$ for $i \in M_k$; copy committed tokens and keep $H_k \setminus M_k$ masked
    **end for**
    Execute final action $x_0$ in $\mathcal{E}$, receive reward $r_t$ and next state $s_{t+1}$
    Estimate value $V_\phi(s_t)$ using the first-step hidden state $h^{(0)}$
    Store $(s_t, x_0, \{H_k, M_k\}_{k=1}^K, r_t)$ in $\mathcal{D}$
  **end for**
  Compute Advantages $A_t$ using GAE over collected rewards and values
  # Phase 2: PPO Update
  **for** epoch $= 1 \ldots E$ **do**
    **for** mini-batch sampled from $\mathcal{D}$ **do**
      # Expand batch across $K$ timesteps to compute parallel transitions
      Initialize $\ell_\theta(\tau_t \mid s_t, \mathcal{M}_t) \leftarrow 0$ and $\ell_{\bar{\theta}}(\tau_t \mid s_t, \mathcal{M}_t) \leftarrow 0$
      **for** $k = K$ **down to** $1$ **do**
        Reconstruct $x_k$ from $x_0$ by masking the recorded positions $H_k$
        Replay the recorded newly decoded positions $M_k$
        Compute $p_\theta(\cdot|x_k, s_t)$ and $p_{\bar{\theta}}(\cdot|x_k, s_t)$
        # The recorded deterministic schedule is fixed for both policies
        $\ell_\theta \mathrel{+}= \sum_{i \in M_k} \log p_\theta(x_0^i \mid x_k, s_t)$
        $\ell_{\bar{\theta}} \mathrel{+}= \sum_{i \in M_k} \log p_{\bar{\theta}}(x_0^i \mid x_k, s_t)$
      **end for**
      # Calculate PPO Objective
      $\widehat{\rho}_t(\theta) = \exp(\ell_\theta(\tau_t \mid s_t, \mathcal{M}_t) - \ell_{\bar{\theta}}(\tau_t \mid s_t, \mathcal{M}_t))$
      $L^{\textbf{dVLA-RL}}(\theta) = \mathbb{E}_t \left[\min\left(\widehat{\rho}_t(\theta)A_t, \text{clip}(\widehat{\rho}_t(\theta), 1-\epsilon, 1+\epsilon)A_t\right)\right]$
      $L^{\text{VF}}(\phi) = \mathbb{E}_{s_t}\left[\left(V_\phi(h_{s_t}^{(0)}) - V_t^{\text{target}}\right)^2\right]$
      # Network Updates
      Update $\theta$ by maximizing $L^{\textbf{dVLA-RL}}(\theta)$
      Update $\phi$ by minimizing $L^{\text{VF}}(\phi)$
    **end for**
  **end for**
**end for**

# B SFT BACKBONE AND EVALUATION DETAILS

## B.1 SFT DATA AND TRAINING SETUP.

For fair comparison with existing VLA-RL studies, we use the same scale of SFT demonstrations as SimpleVLA-RL. On LIBERO, the SFT stage uses 500 demonstrations per suite. On RoboTwin 2.0, the SFT stage uses 1,000 demonstrations per task. Different from SimpleVLA-RL, which fine-tunes policies for individual tasks or suites, our LIBERO and RoboTwin 2.0 SFT checkpoints are obtained through multi-task joint training. Thus, the SFT initialization used by **dVLA-RL** is a unified multi-task discrete diffusion VLA policy rather than a separately fine-tuned task-specific policy.

## B.2 MM-ACT* IMPLEMENTATION.

MM-ACT* follows the MM-ACT architecture and training recipe, with several modifications for efficient RL initialization. First, we use action-only SFT, while the original MM-ACT additionally studies multimodal generation objectives. Second, we set the action chunk size to 16 for both LIBERO and RoboTwin 2.0, whereas the original MM-ACT uses a chunk size of 8 on LIBERO. Third, we use a batch size of 128 on LIBERO and 96 on RoboTwin 2.0. Except for the action-only training setting, the action chunk size, and the batch size, all other SFT hyperparameters follow the original MM-ACT setting.

## B.3 CHECKPOINT SELECTION.

On LIBERO, the Object suite converges faster than the other suites. Therefore, we use the epoch-1 SFT checkpoint for LIBERO-Object and the epoch-4 SFT checkpoint for LIBERO-Spatial, LIBERO-Goal, and LIBERO-Long. For RoboTwin 2.0, we use the epoch-4 SFT checkpoint. These SFT checkpoints are used as the initialization for subsequent online RL post-training.

## B.4 SFT CHECKPOINT EVALUATION.

For LIBERO, we evaluate each subtask with 50 trials, resulting in 2,000 trials over the four LIBERO suites. For RoboTwin 2.0, we evaluate the SFT checkpoint under 4-step denoising with 100 trials per task. Each evaluation scenario is generated with an independent scenario seed.

# C LIBERO RL TRAINING DETAILS

This appendix summarizes the reinforcement learning training configuration used for the LIBERO experiments. For all LIBERO suites, we use the same MM-ACT backbone, action representation, RL algorithm configuration, sampling strategy, and optimization hyperparameters. The denoising step number is fixed to 2 for all suites. The only suite-dependent setting is the maximum rollout horizon, which is adjusted according to the task horizon of each benchmark suite.

As shown in Table 4, the LIBERO experiments use a unified RL training setup across all suites. The policy is initialized from the same MM-ACT checkpoint and optimized with the same actor-critic objective, PPO clipping configuration, sampling parameters, and batch settings. Table 5 lists the only suite-specific difference: the maximum rollout horizon. Shorter-horizon suites such as LIBERO-Spatial use a rollout limit of 256 steps, while the long-horizon LIBERO-Long suite uses a larger limit of 512 steps to accommodate multi-stage manipulation tasks.

Figure 4 shows the PPO training curves on the four LIBERO suites. Across LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long, the success rate consistently increases during training and gradually converges to a high-performance region. The EMA-smoothed curves indicate stable policy improvement despite the variance of raw rollout evaluations. Compared with the shorter-horizon suites, LIBERO-Long uses a larger rollout horizon and exhibits a longer training process, which is consistent with the higher complexity of long-horizon multi-stage manipulation tasks.

Table 4: Shared LIBERO RL training hyperparameters. These parameters are kept identical across LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long.

| Parameter | Value |
|---|---|
| Action representation | ee |
| Action dimension | 7 |
| Action chunk size | 16 |
| Denoising steps | 2 |
| Image resolution | $256 \times 256$ |
| Precision | bf16 |
| Global batch size | 512 |
| Micro batch size | 8 |
| Actor learning rate | $3 \times 10^{-5}$ |
| Value head learning rate | $2 \times 10^{-3}$ |
| Optimizer | AdamW |
| Adam $\epsilon$ | $1 \times 10^{-8}$ |
| Adam $\beta_1, \beta_2$ | 0.9, 0.95 |
| Weight decay | 0.01 |
| Gradient clipping norm | 1.0 |
| Critic warmup steps | 50 |
| PPO update epoch | 1 |
| Rollout trajectories per step | 512 |
| Discount factor $\gamma$ | 0.99 |
| GAE $\lambda$ | 0.95 |
| PPO clip ratio | 0.2 |
| Value clipping | 0.2 |
| Advantage normalization | Yes |
| Reward type | chunk-level |
| Log-probability type | chunk-level |
| Use wrist camera | Yes |
| Use proprioception | Yes |

Table 5: Suite-specific rollout horizons for LIBERO RL training.

| Suite | Maximum rollout steps |
|---|---|
| LIBERO-Spatial | 256 |
| LIBERO-Object | 320 |
| LIBERO-Goal | 320 |
| LIBERO-Long | 512 |

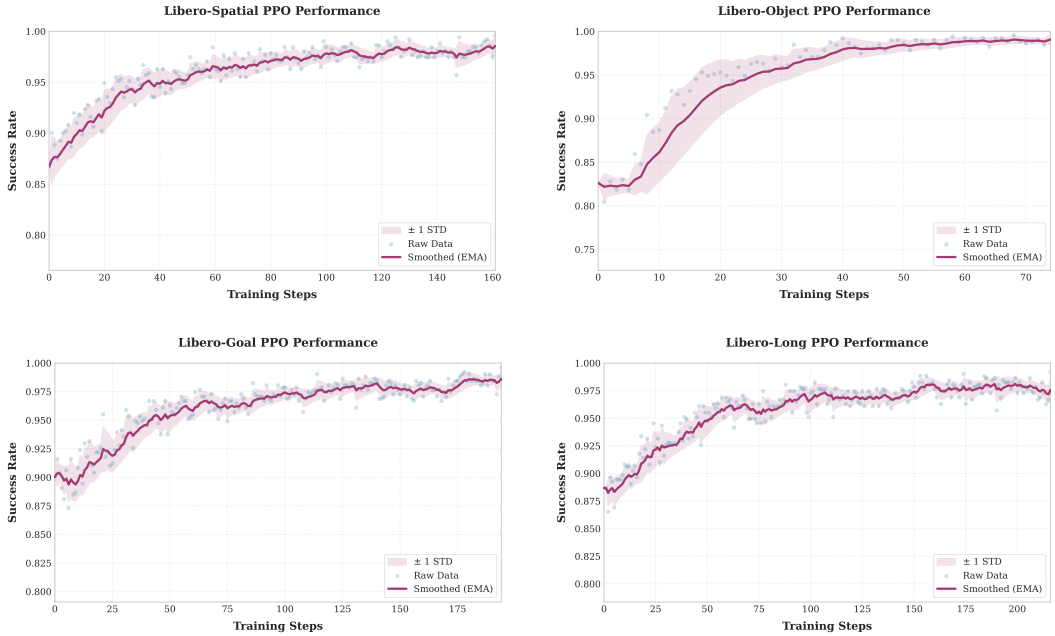



Figure 4: PPO training curves on the LIBERO benchmark suites. Blue points denote raw evaluation results, the smoothed curve denotes the EMA success rate, and the shaded region denotes $\pm 1$ standard deviation.

### C.1 Trajectory-Level Objective Ablation

Section 5.3 studies whether the policy update benefits from modeling the complete denoising trajectory. Unlike a conventional VLA that outputs an action chunk through one prediction, a dVLA conditions each token decision on a partially denoised sequence, and the executable action is obtained only after the full sequence of denoising transitions.

**Necessity of trajectory-level optimization.** We compare the proposed trajectory-level objective with a final-step-only proxy under both two- and four-step denoising. The final-step-only variant retains only the log-probability contribution from the last denoising step and ignores the intermediate transitions that lead to the final action. Because the two variants share the same backbone, rollout setting, sparse outcome reward, and scheduling rule, their difference isolates the effect of using likelihood contributions across the denoising trajectory. Figure 5 shows that the trajectory-level objective produces more stable improvement across tasks and denoising horizons. On LIBERO-Spatial, the final-step proxy improves early but becomes unstable with two-step denoising and plateaus below the trajectory-level objective with four steps. On LIBERO-Object, both variants initially improve, but the proxy later degrades markedly while the trajectory-level objective remains near saturation under both horizons. These results show the empirical benefit of accounting for the complete denoising trajectory rather than relying only on its final transition.

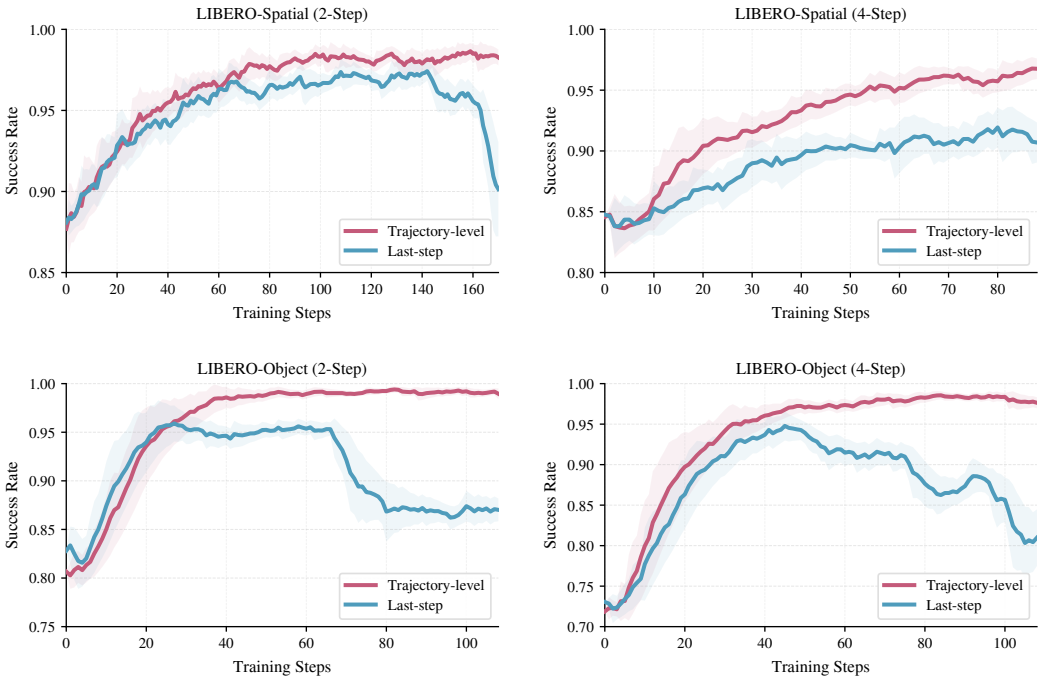



Figure 5: Trajectory-level and final-step-only objectives on LIBERO-Spatial and LIBERO-Object with two or four denoising steps. Curves show EMA success rate; shading denotes $\pm 1$ standard deviation.

## D RoboTwin RL Training Details

This appendix summarizes the reinforcement learning training configuration used for the RoboTwin experiments. To ensure a fair comparison among different denoising-step settings, the 1-step, 2-step, and Hybrid settings use the same training hyperparameters, model configuration, sampling strategy, and environment setup. Their only difference lies in the denoising-step configuration: the 1-step and 2-step settings use a fixed number of denoising steps for all tasks, whereas the Hybrid strategy assigns task-specific denoising steps according to the initial performance and computational cost of each task. For online RL training, each RoboTwin task is associated with a pool of 1,000 distinct environment initialization seeds, which instantiate different training scenarios. At every RL optimization step, we randomly sample 64 seeds from the task-specific pool and collect one rollout trajectory for each sampled seed. Since the training batch contains eight RoboTwin tasks, this produces a total of $64 \times 8 = 512$ rollout trajectories per optimization step. The same seed-sampling protocol is used for all denoising-step variants, ensuring that their differences are not caused by different rollout budgets or environment distributions.

For the fixed-step baselines, all RoboTwin tasks use the same denoising-step number during both rollout and evaluation. Specifically, the 1-step setting uses one denoising step for every task, and the 2-step setting uses two denoising steps for every task. In contrast, the Hybrid setting uses the task-specific denoising-step assignment shown in Table 7. This design preserves the same RL training protocol while allowing the denoising budget to better match the difficulty and efficiency requirement of each task.

Table 6: Shared RoboTwin RL training hyperparameters. These parameters are kept identical for the 1-step, 2-step, and Hybrid settings.

| Parameter | Value |
|---|---|
| Action representation | qpos |
| Action dimension | 14 |
| Action chunk size | 16 |
| Precision | bf16 |
| Global batch size | 512 |
| Micro batch size | 8 |
| Actor learning rate | $3 \times 10^{-5}$ |
| Value head learning rate | $2 \times 10^{-3}$ |
| Optimizer | AdamW |
| Adam $\epsilon$ | $1 \times 10^{-8}$ |
| Adam $\beta_1, \beta_2$ | 0.9, 0.95 |
| Weight decay | 0.01 |
| Gradient clipping norm | 1.0 |
| PPO update epoch | 1 |
| Rollout trajectories per step | 512 |
| Discount factor $\gamma$ | 0.99 |
| GAE $\lambda$ | 0.95 |
| PPO clip ratio | 0.2 |
| Advantage normalization | Yes |
| Reward type | chunk-level |
| Log-probability type | chunk-level |
| Use wrist camera | Yes |
| Use proprioception | Yes |
| Maximum episode steps | 256 |

Table 7: Task-specific denoising steps in the RoboTwin Hybrid setting.

| Task | Denoising steps |
|---|---|
| `beat_block_hammer` | 1 |
| `place_phone_stand` | 1 |
| `place_empty_cup` | 1 |
| `lift_pot` | 2 |
| `move_can_pot` | 2 |
| `place_a2b_left` | 2 |
| `handover_mic` | 4 |
| `pick_dual_bottles` | 4 |

### D.1 Effect of Denoising Steps and Hybrid Scheduling

Section 5.4 analyzes how the denoising-trajectory length affects action generation and RL optimization. The trajectory-level formulation applies to any denoising horizon: one-step decoding is a single parallel transition, two-step decoding introduces one intermediate state, and four-step decoding further unrolls the generation process for finer token refinement. These extra transitions can improve the SFT action chunks and hence the initial success rate before online RL.

A stronger initialization is especially valuable under sparse outcome rewards because successful trajectories appear earlier and provide informative feedback under the same rollout budget. As shown in Figure 3, fixed two-step decoding starts higher and improves faster than fixed one-step decoding during the early and intermediate stages. The Hybrid strategy maintains the highest SR through most of training, reaches the high-success regime earlier, and achieves a peak SR of $0.920$, compared with $0.908$ for fixed two-step and $0.885$ for fixed one-step decoding.

Rather than imposing one horizon on all tasks, the Hybrid strategy assigns one, two, or four steps according to the initial SFT performance, as listed in Table 7. Tasks already solved reliably with fewer steps avoid unnecessary transitions, whereas tasks needing additional refinement retain a longer chain. The denoising horizon therefore affects RL in two coupled ways: it controls action refinement and defines the trajectory structure over which policy probabilities are evaluated and optimized. This task-wise allocation balances initial policy quality, sample efficiency, attainable performance, and compute cost within one unified trajectory-level framework.

### D.2 Training and Inference Efficiency

Fixed multi-step denoising strengthens action refinement but increases both rollout and deployment cost. Within **dVLA-RL**, the Hybrid strategy selects the smallest task-specific horizon whose SFT success rate remains comparable to longer-step decoding, preserving refinement where useful while avoiding unnecessary transitions on easier tasks.

Table 8: Training and inference efficiency under different denoising schedules. Training time is the wall-clock cost of RL training; delay and FLOPs are measured per generated action chunk.

| Setting | Train (s) | Delay (ms) | FLOPs (GFLOPs) |
|---|---|---|---|
| 1-Step | 259.33 | 182.47 | 17547.82 |
| 2-Step | 452.06 | 364.21 | 35095.64 |
| 4-Step | 845.95 | 728.92 | 70191.28 |
| Hybrid | 607.89 | 387.24 | 37289.12 |

Table 8 shows that computation grows nearly proportionally with the denoising horizon. Hybrid requires $607.89$ s of RL training, $387.24$ ms per generated chunk, and $37{,}289.12$ GFLOPs. Relative to fixed four-step decoding, these values reduce training time by $28.1\%$ and both inference delay and FLOPs by $46.9\%$. Although fixed one-step decoding is cheaper, its lower peak SR ($0.885$ versus $0.920$ for Hybrid) shows that minimizing per-step cost alone sacrifices attainable performance. Hybrid therefore offers a stronger performance–efficiency trade-off for large-scale rollout collection and low-latency deployment.

### D.3 RoboTwin Training-Curve Visualizations

Figures 6 and 7 provide the complete training-curve visualizations for the eight RoboTwin tasks. Each plot compares the success-rate evolution of 1-step (PD), 2-step, and the task-adaptive Hybrid strategy under the same RL training protocol.

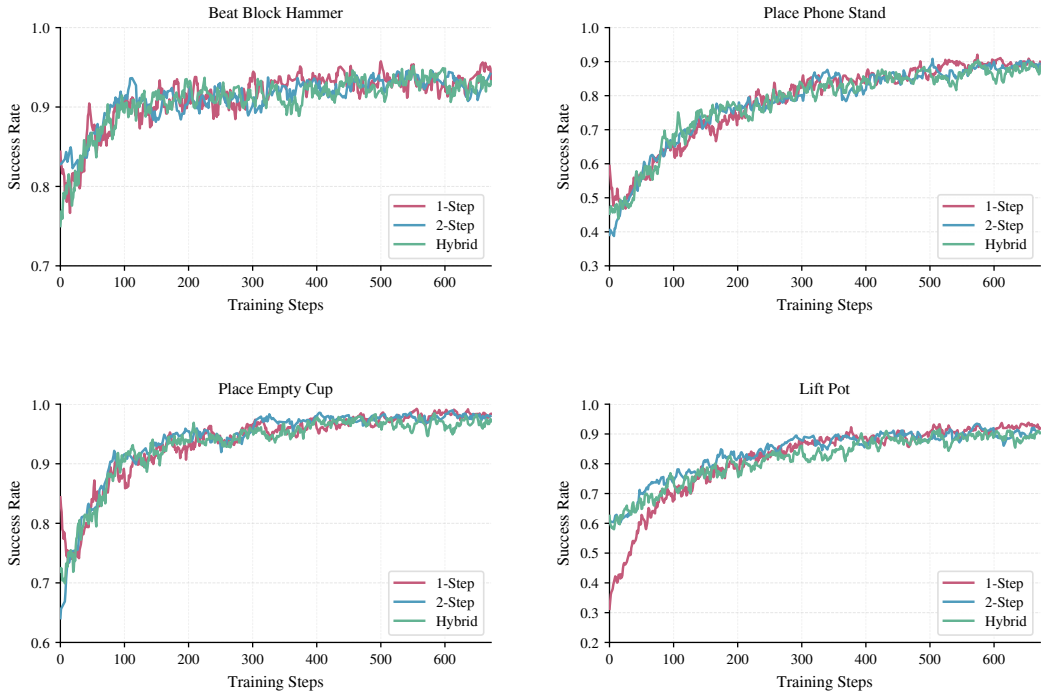



Figure 6: RoboTwin training curves on Beat Block Hammer, Place Phone Stand, Place Empty Cup and Lift Pot. The curves compare 1-Step, 2-Step, and Hybrid in terms of success rate over RL training. For all of 8 tasks, training uses a pool of 1,000 distinct environment initialization seeds, from which 64 seeds are randomly sampled for rollout at every RL optimization step.

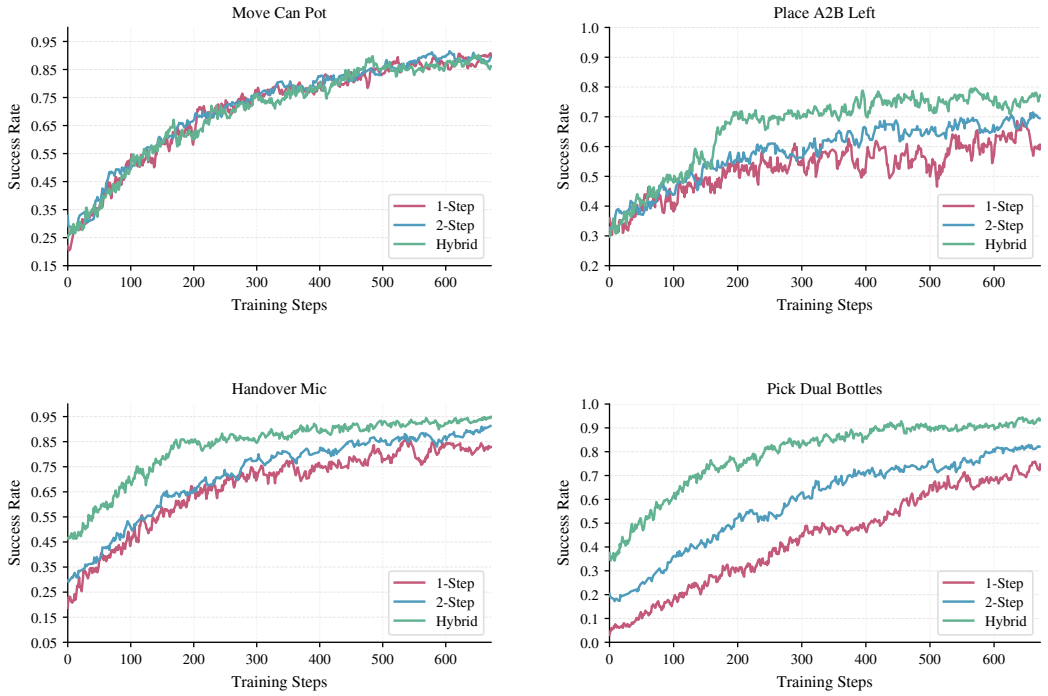



Figure 7: RoboTwin training curves on Move Can Pot, Place A2B Left, Handover Mic, and Pick Dual Bottles, comparing 1-Step, 2-Step, and Hybrid.

# E Real-World Experimental Details

## E.1 Experimental Platform

We conduct the physical experiments on an AgileX Cobot Magic platform configured for ALOHA-style bimanual manipulation (Figure 8). Each working arm is an AgileX Piper manipulator with six actuated joints and a parallel gripper with one additional degree of freedom. Three Intel RealSense D435 RGB-D cameras observe the workspace: a head-mounted camera provides a global view, while a camera at each wrist captures the local gripper–object interaction. The head camera records RGB images at $640 \times 480$ pixels and 30 Hz. This arrangement provides complementary views for object localization, grasping, and coordination between the two arms.




Figure 8: Real-world experimental platform. The AgileX Cobot Magic setup supports ALOHA-style dual-arm manipulation, with one head camera and two wrist cameras observing the shared workspace.

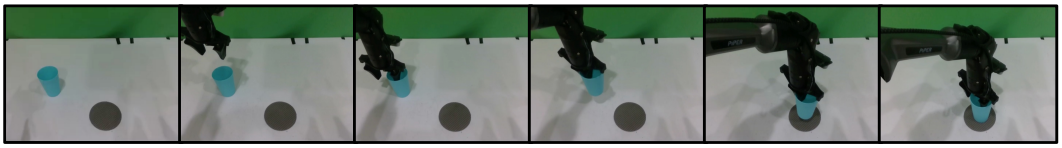
## E.2 Short-Horizon Tasks

The three short-horizon tasks are adapted from RoboTwin to the physical workspace. Figure 9 shows representative execution sequences.

**Place Empty Cup (PEC).** A light blue plastic cup and a round coaster are placed on the table. The robot grasps the cup, moves it above the coaster, and lowers and releases it onto the coaster. The task requires aligning the cup with a small placement target while maintaining a stable grasp during transport. The language instruction is *"Use arm to place the light blue plastic cup on the round coaster"*.

**Lift Pot (LP).** A pot initially covers the round coaster. The robot grasps the two side handles with its two arms, lifts the pot, and moves it forward before setting it down, leaving the coaster exposed. Coordinating the two grasps and the subsequent motion is necessary to move the pot without tilting it excessively. The language instruction is *"Uncover the round coaster by lifting the pot and placing it forward."*

**Handover Mic (HM).** A microphone lies on the table. One arm picks it up and presents it to the other arm; the receiving gripper secures the microphone before the first gripper releases it. This task requires coordinated positioning and gripper timing during the transfer. The language instruction is *"Grasp the microphone with rounded mesh head, transfer it, then let go of it smoothly."*



(a) Place Empty Cup: grasp the cup, transport it to the coaster, and place it on the target.



(b) Lift Pot: grasp both handles and move the pot forward to uncover the coaster.



(c) Handover Mic: pick up the microphone and transfer it between the two grippers.

Figure 9: Representative real-world executions of the three short-horizon tasks. Frames in each row are ordered from left to right.

### E.3 Long-Horizon Task: Organize Table

Organize Table (OT) combines three ordered manipulation stages under the language instruction *"Organize the objects on the table."* The initial scene contains a carrot on a plate, a pot covering a round coaster, and a light blue cup beside the pot. The robot first transfers the carrot from the plate into the pot (Pick and Place Carrot, PPC). It then grasps the pot with both arms and moves it forward to expose the coaster (LP). Finally, it picks up the cup and places it onto the exposed coaster (PEC). Figure 10 illustrates the full sequence.

We introduce this long-horizon task to examine the robustness of learned action patterns to environmental noise and deviations during task execution. The stages share objects and depend on the preceding actions: the pot is moved with the carrot inside, and the cup placement target becomes accessible only after the pot has been relocated. Intermediate object and gripper configurations carry over between stages, testing sustained execution across single-arm pick-and-place actions and coordinated bimanual manipulation.

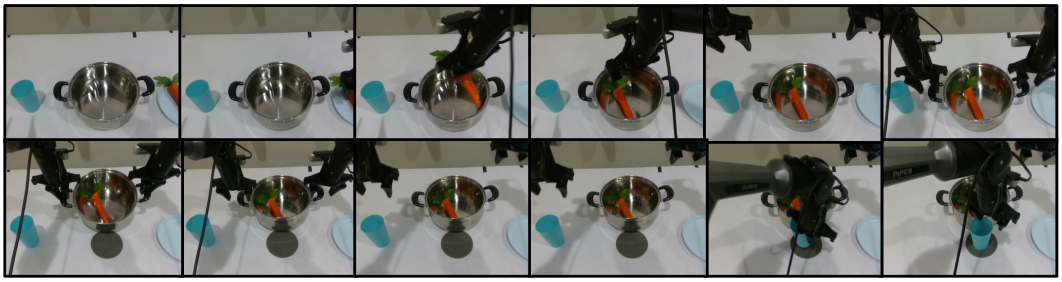


Figure 10: Representative execution of Organize Table. Frames proceed from left to right across the top row and then the bottom row, showing carrot transfer into the pot, bimanual pot relocation to reveal the coaster, and cup placement onto the coaster.

### E.4 Demonstrations, Adaptation, and Evaluation

**Demonstration datasets and joint training.** We collect 100 demonstration trajectories for each of PEC, LP, and HM. For short-horizon evaluation, we jointly train on the combined set of 300 demonstrations; the reported success rates for these three tasks are obtained from the resulting multi-task checkpoint for each method. For long-horizon evaluation, we add 150 Organize Table demonstrations to the same 300 short-horizon demonstrations and jointly train on all 450 trajectories. The OT stage success rates are evaluated using the checkpoints trained on this four-task mixture. Thus, the short-horizon and OT columns in Table 3 correspond to the 300- and 450-demonstration training settings, respectively.

**Initialization and real-world adaptation.** For **dVLA-RL**, real-world SFT starts from the RoboTwin Hybrid checkpoint at RL training step 700. MM-ACT* and $\pi_{0.5}$ start from their respective RoboTwin SFT checkpoints, specifically those corresponding to the success rates reported in Table 2. All three methods are adapted through SFT on the real-world demonstration mixtures described above. No RL is performed on the physical robot; the comparison between **dVLA-RL** and MM-ACT* measures the benefit of the RL-trained initialization after real-world SFT.

**Real-world SFT and execution configuration.** All three methods—**dVLA-RL**, MM-ACT*, and $\pi_{0.5}$—use a batch size of 128 and an action chunk size of 16 during real-world SFT. At inference, each method executes all 16 actions in the predicted chunk before generating the next chunk. For **dVLA-RL** and MM-ACT*, the remaining SFT settings follow the RoboTwin SFT configuration in Appendix B, including action-only supervision. For $\pi_{0.5}$, all other training hyperparameters follow the $\pi_{0.5}$ training configuration provided by RoboTwin. The real-world task mixtures are specified above.

**Success criteria.** We assess each short-horizon task by its completed manipulation outcome:

- **PEC:** the arm picks up the cup and places it onto the coaster.
- **LP:** both arms lift the pot, move it forward, and set it down so that the previously covered coaster is exposed.
- **HM:** one arm picks up the microphone and passes it to the other; after the receiving gripper grasps it and the original gripper opens, the receiving arm continues to hold the microphone.

For OT, the three stages are scored separately: (1) the robot picks up the carrot and places it inside the pot; (2) it lifts and sets down the pot, exposing the previously covered coaster; and (3) it picks up the cup and places it onto the coaster.

**Reported metrics.** Table 3 reports percentages, with three subcolumns for the stages of OT. The aggregate OT score used in the overall average is the arithmetic mean of these three stage success rates. The overall average assigns equal weight to PEC, LP, HM, and this OT score, and therefore is not an average of four full-task completion rates. All averages and differences are calculated from the reported percentages before rounding to one decimal place.