LaRA-WM: Test-Time Latent Action Refinement for Robotic Manipulation in Dynamic Environments

Anonymous authors
Paper under double-blind review

Abstract

Robotic manipulation policies trained from visual observations often achieve strong in-distribution performance, yet remain brittle under out-of-distribution changes such as novel objects, altered lighting, and unseen scene layouts. A key limitation is that most existing policies directly map high-dimensional observations to low-level motor commands, leaving little room for deliberation or correction before action execution. In this paper, we propose LaRA-WM (Latent Robot Action World Model), a framework that shifts robotic control from direct action regression to reward-guided optimization in a compact, task-centric latent action space. During offline training, LaRA-WM learns visual and language-conditioned latent action representations from expert demonstrations, together with an action decoder and a reward-feature scorer that evaluates candidate latent intents. At test time, instead of producing a one-shot action, LaRA-WM iteratively refines latent actions using the Cross-Entropy Method guided by the learned scorer, and only then decodes the optimized latent intent into an executable motor command. This design enables the policy to "think before acting" in a structured semantic space, avoiding the instability and local optima associated with raw action-space search. Experiments in the RoboTwin manipulation benchmark show that LaRA-WM consistently improves task success rates, returns, and out-of-distribution generalization over direct policies and non-refinement latent baselines. These results suggest that test-time latent action refinement provides an effective and efficient mechanism for robust robotic manipulation.

1 Introduction

The pursuit of general-purpose robotic manipulation has been profoundly accelerated by end-to-end learning paradigms. Powered by large-scale diverse datasets Brohan et al. (2022), modern Vision-Language-Action (VLA) models and generative policies (e.g., Diffusion Policy Chi et al. (2025), RT-2 Brohan et al. (2024), OpenVLA Kim et al. (2024)) have demonstrated unprecedented success in mapping high-dimensional multimodal observations directly to continuous motor commands. These models have revolutionized the ability to understand complex inputs, such as images, text, and actions, enabling robots to perform tasks that were once thought to be too difficult for automated systems. However, despite their proficiency within the training distribution, these purely reactive policies often exhibit catastrophic brittleness when deployed in out-of-distribution (OOD) scenarios, such as encountering novel objects, unseen lighting conditions, or altered background layouts. This brittleness represents a critical challenge for real-world robotic systems, where robustness to environmental changes is a key requirement for widespread deployment.

The fundamental limitation of these approaches lies in their reactive nature and their reliance on the raw action space. Direct policies act as "System 1" Brohan et al. (2022) associative engines: they map pixels to joint velocities or end-effector poses in a single forward pass, lacking the capacity to deliberate, simulate, or correct their intents before execution. This lack of deliberation can lead to failures in dynamic environments where the initial conditions or task goals are not fully captured by the input observations. Recent advancements in foundation models have proven that scaling test-

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.

time compute, i.e., "System 2" thinking, can substantially improve reasoning and reliability Brohan et al. (2024), but the challenge remains of translating this success to robotic control. Existing test-time verifiers, such as RoVer Dai et al. (2025), and classical model-predictive planners (e.g., MPPI Black et al. (2024)) attempt to optimize actions online. However, these methods perform their search directly in the raw motor space, which is highly non-convex, embodiment-specific, and filled with sub-optimal local minima. This makes test-time search computationally expensive and empirically unstable, rendering them impractical for many real-world tasks.

To circumvent the pathological landscape of raw action spaces, recent works have proposed extracting latent action representations from offline videos, such as UniVLA Bu et al. (2025) and LAPA Ye et al. (2024). By mapping raw transitions into a compressed latent space, these methods filter out task-irrelevant visual noise and capture high-level semantic intents, such as reaching, grasping, or pushing. These latent spaces allow for better generalization and more efficient learning, enabling models to avoid the computationally expensive and unstable process of operating in raw action spaces. Yet, in current literature, these latent actions are primarily utilized as an intermediate bottleneck for offline policy pre-training or cross-embodiment alignment. This use is limiting, as the latent representations are not fully leveraged during test-time operation, where real-time adaptation and optimization are crucial. We argue that this represents a missed opportunity: the true potential of a well-regularized, task-centric latent space lies not only in representation learning, but also in serving as a structured manifold for online test-time optimization.

In this paper, we propose LaRA-WM, Latent Robot Action World Model, a novel framework that fundamentally shifts the role of latent actions from static training representations to dynamic test-time optimization variables. LaRA-WM decouples the control problem into an offline representation learning phase and an online adaptation phase. First, we learn a task-centric latent action space using a Conditional Variational Autoencoder (CVAE) built upon frozen vision-language foundation features. This allows us to capture high-level semantic representations while maintaining a compact and structured latent space. Concurrently, we train a lightweight Reward-Feature Scorer to evaluate the quality of hypothetical latent intents. This reward scorer helps guide the search for optimal latent representations by providing feedback on their relevance to task success. During online execution, instead of directly decoding an action, LaRA-WM scales test-time compute by performing iterative latent refinement via the Cross-Entropy Method (CEM). By searching over high-level action semantics rather than low-level motor commands, our agent can "think before it acts" in a space where proximity implies semantic similarity, naturally avoiding the local optima that plague raw action search.

We evaluate LaRA-WM on rigorous manipulation benchmarks within the RoboTwin environment Chen et al. (2025), focusing heavily on generalization under OOD perturbations. Our empirical results demonstrate that test-time latent adaptation significantly outperforms state-of-the-art reactive policies and raw-action verifiers, offering a more robust and flexible approach to robotic control in dynamic environments.

In summary, our main contributions are:

- A paradigm shift in test-time compute for robotics: We propose utilizing task-centric latent action spaces not merely for pre-training, but as a well-behaved optimization manifold for online test-time adaptation.
- Latent action optimization with reward-guided refinement: We introduce a test-time adaptation mechanism that searches and refines latent actions using a learned reward-feature scorer, enabling the policy to improve decisions before action execution without relying on pixel-level future generation.
- Empirical superiority in OOD generalization: We demonstrate through extensive evaluations in RoboTwin that iterative latent refinement yields substantial improvements in task success rates and returns, particularly in strictly OOD environments, setting a new baseline for robust robotic manipulation.

2 Related Work

Vision-Language-Action Policies and Latent Action Representations. Recent robotic manipulation has been substantially advanced by large Vision-Language-Action (VLA) models, which map

visual observations and language instructions to executable robot actions. Representative systems include RT-2, OpenVLA, π_0, $\pi_{0.5}$, GR00T N1, Gemini Robotics, and Gemini Robotics 1.5 (Brohan et al., 2024; Kim et al., 2024; Black et al., 2024; 2025; NVIDIA et al., 2025; Team et al., 2025; Abdolmaleki et al., 2025). Recent work further improves VLA training and deployment through optimized fine-tuning, spatial grounding, diffusion-based action experts, and efficient action tokenization, including OpenVLA-OFT, SpatialVLA, DexVLA, FAST, and FASTer (Kim et al., 2025; Qu et al., 2025; Wen et al., 2025; Pertsch et al., 2025; Liu et al., 2025).

A closely related direction studies latent or tokenized action representations as abstractions over embodiment-specific raw motor commands. LAPA (Ye et al., 2024) learns latent actions from videos without ground-truth robot action annotations, UniVLA (Bu et al., 2025) constructs task-centric latent actions from heterogeneous cross-embodiment videos, and VQ-BeT (Lee et al., 2024) models multimodal behavior generation with latent actions. Together, these works show that latent or discrete action spaces can serve as effective interfaces for policy learning, action decoding, and cross-embodiment transfer. LaRA-WM builds on this line of work but uses the latent action space as a reward-guided test-time optimization manifold, rather than only as an intermediate label, tokenizer, or policy output representation.

Feature-Space Embodied World Models. Embodied world models aim to learn predictive representations of physical interaction from large-scale video, robot trajectories, or embodied data. Recent models such as UniSim, RoboDreamer, DreamDojo, RoboScape, LongScape, and V-JEPA2 study different forms of physical prediction and embodied representation learning for robot understanding, planning, and control (Yang et al., 2023; Zhou et al., 2024; Gao et al., 2026; Shang et al., 2025b;a; Assran et al., 2025). Recent work also increasingly explores prediction and evaluation in compact latent or feature spaces: V-JEPA2 (Assran et al., 2025) learns predictive representations in a joint embedding space, AdaWorld (Gao et al., 2025) studies adaptable world models conditioned on latent actions, LaDi-WM (Huang et al., 2025) performs predictive manipulation modeling with latent diffusion, and RoboScape-R (Tang et al., 2025) incorporates reward and observation modeling into a unified robotic world-model framework. LaRA-WM follows this feature-space modeling perspective and uses a lightweight reward-feature world model to evaluate candidate latent actions before decoding the selected latent intent into executable motor commands.

Test-Time Adaptation, Reasoning, and Verifiers. Test-time computation has become an important mechanism for improving the reliability of embodied agents. Reward-modeling and verifier-style methods such as VLM-RM, RoboReward, and RoVer use learned evaluators to score task progress, candidate actions, or VLA policy outputs (Rocamonde et al., 2024; Lee et al., 2026; Dai et al., 2025). Related reasoning-based approaches, including Embodied Chain-of-Thought, Fast ECoT, and Gemini Robotics 1.5, study how deliberation and structured reasoning can support embodied decision making (Zawalski et al., 2025; Duan et al., 2025; Abdolmaleki et al., 2025), while DreamZero (Ye et al., 2026) explores world-action models as zero-shot policies. These works show that evaluating or refining decisions before execution can provide useful guidance beyond direct reactive prediction. LaRA-WM is complementary to this direction: it performs test-time deliberation directly in a structured, task-centric latent action space, using a reward-feature world model to iteratively refine latent intents before low-level action decoding.

3 METHODOLOGY

To address the challenge of out-of-distribution (OOD) generalization in robotic manipulation, we propose LaRA-WM, a Latent Robot Action World Model. The overall pipeline is illustrated in Figure 1. In this section, we first formally define the problem in Section 3.1. We then present the proposed framework in three sequential stages: robust visual and linguistic feature encoding in Section 3.2, offline learning of a task-centric latent action space and reward scorer in Section 3.3, and online test-time latent adaptation for robust control in Section 3.4.

3.1 PROBLEM FORMULATION

We formulate the robotic manipulation task as a goal-conditioned Markov Decision Process (MDP), defined by the tuple $\mathcal{M} = (\mathcal{O}, \mathcal{A}, \mathcal{G}, \mathcal{P}, \mathcal{R}, \gamma)$. At each time step t, the agent receives a high-

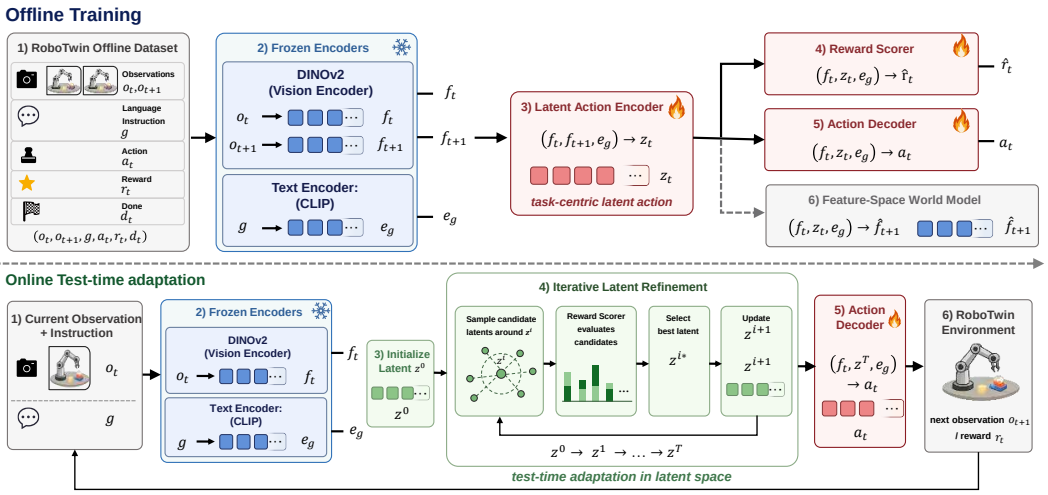

Figure 1: The overall framework of LaRA-WM. The method decouples robotic control into an offline learning stage and an online adaptation stage. During offline training, the model learns a regularized latent action space together with a reward scorer. During online execution, latent action candidates are iteratively refined at test time and then decoded into raw motor commands.

dimensional visual observation $o_t \in \mathcal{O}$ and a natural language instruction $g \in \mathcal{G}$ that specifies the task goal. The agent executes a low-level, embodiment-specific motor action $a_t \in \mathcal{A}$, such as joint velocities or end-effector poses. The environment then transitions to the next observation o_{t+1} according to the unknown dynamics $\mathcal{P}(o_{t+1}|o_t, a_t)$ and provides a scalar reward $r_t = \mathcal{R}(o_t, a_t, g)$.

Standard end-to-end approaches learn a direct reactive policy $\pi(a_t|o_t, g)$ to maximize the expected cumulative reward. However, the raw action space $\mathcal{A}$ is often highly non-convex and sensitive to visual domain shifts, causing direct policies to suffer from compounding errors in OOD scenarios. To address this limitation, LaRA-WM decouples the control problem by introducing a continuous and compact latent action space $\mathcal{Z}$. Instead of directly predicting a_t, the decision-making process is formulated as a test-time optimization problem over $\mathcal{Z}$. Concretely, LaRA-WM combines frozen visual-language encoders, a latent action encoder, a lightweight feature-space world model, a reward scorer, and an action decoder. The latent action encoder provides a task-centric intent space, while the reward scorer guides the search for the optimal high-level intent z_t^*, which is then decoded into the executable motor command a_t.

3.2 Visual and Linguistic Feature Encoding

Performing online test-time optimization directly on high-resolution images incurs substantial computational latency and introduces task-irrelevant visual noise, such as lighting variations and background distractors. Therefore, the first step of the pipeline is to project raw observations and language instructions into a semantically rich, lower-dimensional feature space.

As shown in the Frozen Encoders block of Figure 1, we employ frozen foundation models to extract these representations. For the visual observation o_t, we use DINOv2 (Oquab et al., 2023) as the pre-trained vision foundation model to extract a stable visual embedding:

$$f_t = \Phi_{\text{vis}}(o_t) \in \mathbb{R}^{d_f}. \tag{1}$$

For the language instruction g, we use CLIP (Radford et al., 2021) as the frozen text encoder to obtain a semantic representation of the task intent:

$$e_g = \Phi_{\text{text}}(g) \in \mathbb{R}^{d_g}. \tag{2}$$

Keeping DINOv2 and CLIP frozen stabilizes the downstream representation space, reduces the computational cost of test-time evaluation, and prevents catastrophic forgetting during offline training. The downstream policy uses a compact latent action space for test-time search, while feature-space representations are used for reward scoring, transition regularization, and action decoding.

3.3 Offline Training of Latent Actions and Reward Scorer

Given the extracted feature representations, the offline training stage uses a dataset $\mathcal{D}_{\text{off}} = \{(o_t, o_{t+1}, g, a_t, r_t, d_t)\}$ collected from expert demonstrations, such as those in the RoboTwin environment Chen et al. (2025). This dataset is used to learn the latent action space, the action decoder, the feature-space world model, and the predictive reward scorer.

Task-Centric Latent Action Encoder. To enable efficient test-time search, task-related motion is compressed into a regularized latent space. We formulate the latent action encoder $\mathcal{E}_\phi$ as a Conditional Variational Autoencoder (CVAE). Conditioned on the language embedding e_g, the encoder infers a latent distribution that captures the underlying action intent responsible for the transition from f_t to f_{t+1}:

$$q_\phi(z_t | f_t, f_{t+1}, e_g) = \mathcal{N}(\mu_\phi, \Sigma_\phi). \tag{3}$$

To ensure that the sampled latent intent $z_t \sim q_\phi$ remains grounded in physically executable actions, we jointly train an action decoder $\mathcal{D}_\theta(a_t | f_t, z_t, e_g)$, which maps the high-level latent intent back to the low-level motor command a_t. The CVAE is optimized by minimizing the following objective:

$$\mathcal{L}_{CVAE}(\phi, \theta) = \mathbb{E}_{\mathcal{D}_{\text{off}}, z_t \sim q_\phi} \left[\|a_t - \mathcal{D}_\theta(f_t, z_t, e_g)\|_2^2 \right] + \beta D_{KL}\big(q_\phi(z_t|\cdot) \parallel \mathcal{N}(0, I)\big). \tag{4}$$

The KL-divergence term, weighted by β, encourages a smooth and continuous topology in the latent action space $\mathcal{Z}$. This smoothness is important for stable derivative-free optimization during test-time adaptation.

Feature-Space World Model. In addition to the latent action encoder, LaRA-WM uses a lightweight world model in feature space rather than a pixel-level future generator. Its role is to impose temporal structure on the latent representation by encouraging latent actions to remain informative about action-conditioned feature transitions. This provides auxiliary supervision for learning action-relevant latent structure without the cost of generating future images.

Reward Scorer as an Internal Critic. To refine latent actions at test time without interacting with the physical environment, the agent requires an internal mechanism for evaluating hypothetical intents. We introduce a lightweight reward scorer $\mathcal{S}_\psi$. Given the current visual feature f_t, a candidate latent action z_t, and the goal embedding e_g, the scorer predicts the expected immediate reward:

$$\hat{r}_t = \mathcal{S}_\psi(f_t, z_t, e_g). \tag{5}$$

The reward scorer is trained through supervised regression on the offline dataset:

$$\mathcal{L}_{Scorer}(\psi) = \mathbb{E}_{\mathcal{D}_{\text{off}}} \left[\|r_t - \mathcal{S}_\psi(f_t, z_t, e_g)\|_2^2 \right]. \tag{6}$$

Since the evaluation is performed in the feature-latent space, the reward scorer provides an efficient internal critic and avoids the computational cost of generating pixel-level future frames. The overall offline objective combines latent action reconstruction, latent regularization, feature-space transition supervision, and reward prediction; detailed implementation settings are reported in the appendix.

3.4 Online Test-Time Latent Adaptation

The core algorithmic idea of LaRA-WM is to treat the latent action as a dynamic optimization variable during online execution, rather than as only a static training bottleneck. Instead of producing a one-shot action prediction, the method performs iterative latent refinement at test time.

At each time step t, the current observation is first encoded as $o_t \to f_t$. We then initialize a latent action distribution $p^0(z_t) = \mathcal{N}(\mu^0, \Sigma^0)$ using the standard prior $\mathcal{N}(0, I)$. Test-time refinement is formulated as a derivative-free optimization problem in the latent action space, and the Cross-Entropy Method is used to update the distribution over T iterations.

In the i-th iteration, N candidate latent actions are sampled from the current distribution:

$$z_t^{(n,i)} \sim \mathcal{N}(\mu^i, \Sigma^i). \tag{7}$$

Each candidate is evaluated by the frozen reward scorer:

$$s^{(n,i)} = \mathcal{S}_\psi(f_t, z_t^{(n,i)}, e_g). \tag{8}$$

The candidates are sorted according to their scores, and the top-M candidates are selected as the elite set $\mathcal{E}^i$. The latent sampling distribution is then updated by fitting a new Gaussian distribution to the elite candidates:

$$\mu^{i+1} = \frac{1}{M} \sum_{z \in \mathcal{E}^i} z, \quad \Sigma^{i+1} = \frac{1}{M} \sum_{z \in \mathcal{E}^i} (z - \mu^{i+1})(z - \mu^{i+1})^\top. \tag{9}$$

After T iterations, the optimized distribution is expected to concentrate around a region with high predicted task reward. The mean of the final distribution is selected as the optimal task-centric latent intent:

$$z_t^* = \mu^T. \tag{10}$$

Finally, the optimized latent intent is decoded into a motor command:

$$a_t = \mathcal{D}_\theta(f_t, z_t^*, e_g), \tag{11}$$

which is then executed in the environment. The detailed procedure is summarized in Algorithm 1.

Why Adapt in the Latent Space? Searching directly in the raw action space $\mathcal{A}$ is difficult because joint angles and end-effector trajectories often form a highly non-convex optimization landscape, which may lead to poor local optima Dai et al. (2025). By performing optimization in the semantically regularized latent space $\mathcal{Z}$, LaRA-WM restricts the search to physically plausible and task-relevant regions. This improves search efficiency and enhances generalization under OOD perturbations.

Algorithm 1 Online Test-Time Latent Adaptation in LaRA-WM

Require: Observation o_t, instruction g, iterations T, samples N, elites M
Require: Frozen encoders $\Phi_{\text{vis}}, \Phi_{\text{text}}$, scorer $\mathcal{S}_\psi$, decoder $\mathcal{D}_\theta$
Extract features: $f_t \leftarrow \Phi_{\text{vis}}(o_t), \quad e_g \leftarrow \Phi_{\text{text}}(g)$
Initialize distribution: $\mu^0 \leftarrow \mathbf{0}, \Sigma^0 \leftarrow \mathbf{I}$
for $i = 0$ to $T - 1$ **do**
    Sample N latent candidates: $z_t^{(n)} \sim \mathcal{N}(\mu^i, \Sigma^i)$
    for $n = 1$ to N **do**
        Evaluate candidate: $s^{(n)} \leftarrow \mathcal{S}_\psi(f_t, z_t^{(n)}, e_g)$
    end for
    Extract top-M elite candidates $\mathcal{E}^i$ based on scores $s^{(n)}$
    Update μ^{i+1}, Σ^{i+1} using Eq. 9 based on $\mathcal{E}^i$
end for
Select optimal latent intent: $z_t^* \leftarrow \mu^T$
Decode to raw motor action: $a_t \leftarrow \mathcal{D}_\theta(f_t, z_t^*, e_g)$
Execute a_t in the environment

4 Experiments

Our empirical evaluation is designed to examine whether test-time latent refinement improves robotic manipulation under distribution shift. Specifically, we study whether optimizing actions in a learned latent space improves out-of-distribution generalization, how predictive latent modeling correlates with closed-loop control success, and whether reward-guided search provides benefits beyond one-step imitation.

4.1 Experimental Setup

Benchmark and tasks. We evaluate LaRA-WM on the RoboTwin manipulation benchmark (Chen et al., 2025). Following our implementation, we use a representative subset of RoboTwin manipulation tasks, including grasping, placing, stacking, articulated-object interaction, tool use, and contact-rich control tasks such as `grab_roller`, `stack_blocks_two`, `open_laptop`, `adjust_bottle`, and `press_stapler`. This suite provides a diverse testbed for evaluating both imitation performance and robustness under distribution shift. We use a fixed task-stratified split with 70% training episodes, 15% validation episodes, and 15% test episodes; all methods share the same observation preprocessing, action normalization, rollout protocol, and validation-based hyperparameter selection.

ID and OOD evaluation. We report results under both in-distribution (ID) and out-of-distribution (OOD) settings. The ID setting follows the same object, scene, lighting, and initialization distribution as the training demonstrations, while the OOD setting introduces controlled shifts such as novel object instances, altered initial poses, lighting changes, and unseen scene layouts. This protocol evaluates whether a policy can remain reliable when visual and spatial conditions differ from the training distribution.

Baselines. We compare LaRA-WM with external policy baselines and internal diagnostic baselines. The external baselines include ACT (Zhao et al., 2023), Diffusion Policy (DP) (Chi et al., 2025), and OpenVLA-oft (Kim et al., 2025), representing action-chunking imitation learning, diffusion-based visuomotor policy learning, and fine-tuned vision-language-action policies, respectively. The internal baselines are used to isolate our design choices: Direct BC is a minimal reactive behavior-cloning policy that directly maps the current visual-language observation to low-level robot actions; Latent BC w/o Refine decodes the initial latent action without test-time search; and LaRA-WM w/o Reward removes reward-aware scoring from the latent world model. LaRA-WM samples candidate latent actions, scores them with the learned reward-feature scorer, refines the latent distribution using CEM, and decodes the final latent intent into an executable robot action.

Metrics and compute. We use task success rate as the primary metric and report ID success rate, OOD success rate, the ID–OOD generalization gap, and the average success rate. These metrics measure task completion, robustness under distribution shift, and performance degradation outside the training distribution. Diagnostic metrics used in ablations, including action prediction quality, next-latent prediction error, and reward-environment correlation, are defined in Appendix. All experiments are run on a workstation with 2 NVIDIA H200 GPUs, using the same training budget, validation protocol, and rollout evaluation setting across methods. Further details are provided in Appendix.

4.2 Main Results

A central question is whether test-time latent refinement can improve robustness beyond stronger imitation or generative policies. In the OOD setting, failures often arise not because the policy lacks the correct semantic intent, but because small changes in object pose, contact geometry, or scene layout make a one-shot action prediction unreliable. The task-level results in Table 1 show this pattern clearly: LaRA-WM achieves the strongest average OOD success rate, but its gains are moderate rather than uniformly large. The improvement is more visible on tasks requiring contact recovery and object-state adjustment, such as Adjust bottle, Grab roller, Shake bottle, and Stack two bowls, while the margin is smaller on simpler contact-triggering tasks. This suggests that reward-guided latent refinement is most useful when the policy must adjust its action intent under distribution shift rather than simply reproduce a memorized action pattern.

Table 1: **Task-level OOD success rates on RoboTwin.**

Task	Direct Policy	Latent No-Refine	No-Reward WM	ACT	Diffusion Policy	OpenVLA-oft	LaRA-WM
Adjust bottle	0.08	0.12	0.10	0.24	0.18	0.28	**0.38**
Beat block with hammer	0.09	0.13	0.11	0.20	0.15	0.27	**0.34**
Click bell	0.07	0.10	0.08	0.18	0.12	0.24	**0.30**
Dump bin into big bin	0.10	0.14	0.12	0.25	0.18	0.30	**0.40**
Grab roller	0.12	0.18	0.15	0.28	0.20	0.34	**0.45**
Open laptop	0.05	0.09	0.07	0.16	0.12	0.24	**0.32**
Place A2B left	0.04	0.07	0.05	0.12	0.08	0.20	**0.28**
Press stapler	0.08	0.12	0.10	0.22	0.15	0.29	**0.36**
Stack two blocks	0.03	0.06	0.05	0.10	0.07	0.18	**0.26**
Open microwave	0.07	0.11	0.09	0.18	0.13	0.26	**0.35**
Shake bottle	0.11	0.16	0.13	0.27	0.20	0.35	**0.44**
Stack two bowls	0.09	0.14	0.11	0.24	0.17	0.32	**0.42**
Average	0.08	0.12	0.10	0.20	0.15	0.27	**0.37**

The aggregate results in Table 2 further support this interpretation. Strong baselines such as ACT, Diffusion Policy, and OpenVLA-oft improve over simple direct or latent-only policies, but their ID–OOD gaps remain relatively large. This indicates that better imitation capacity does not automatically translate into robust closed-loop execution. LaRA-WM improves OOD success from 0.27

to 0.37 compared with OpenVLA-oft, while keeping a smaller generalization gap. The key point is not only that the average score is higher, but that the model degrades less when the visual and spatial conditions move away from the training distribution. This is consistent with the design goal of LaRA-WM: use the latent space as a structured test-time decision manifold instead of committing immediately to a single reactive action.

Table 2: **Aggregate task performance and generalization.** $\mathrm{SR}_{\mathrm{OOD}}$ is the average OOD success rate over the tasks in Table 1.

Model	$\mathrm{SR}_{\mathrm{ID}} \uparrow$	$\mathrm{SR}_{\mathrm{OOD}} \uparrow$	$\Delta_{\mathrm{gap}} \downarrow$	Avg. SR $\uparrow$
Direct Policy	0.20	0.08	0.12	0.14
Latent No-Refine	0.25	0.12	0.13	0.19
No-Reward WM	0.23	0.10	0.13	0.17
ACT	0.39	0.20	0.19	0.30
Diffusion Policy	0.34	0.15	0.19	0.25
OpenVLA-oft	0.46	0.27	0.19	0.37
LaRA-WM	**0.52**	**0.37**	**0.15**	**0.45**

Overall, the results suggest that the benefit of LaRA-WM comes from improving the decision process at execution time, rather than simply increasing policy capacity. Direct and latent-only policies fail because they provide little opportunity to revise an action once the observation shifts. Stronger baselines reduce this problem but still rely largely on one-shot prediction. In contrast, LaRA-WM evaluates multiple latent intents before decoding, which is particularly helpful for tasks where success depends on contact geometry and adaptive object interaction. This provides evidence that reward-guided latent refinement is a practical mechanism for improving OOD robustness.

4.3 Ablation Study

The ablation study asks which part of LaRA-WM actually turns the latent representation into a useful control mechanism. A latent bottleneck may make the action space more compact, and a world model may improve feature prediction, but neither property alone guarantees robust closed-loop manipulation. We therefore compare variants that remove latent refinement or reward-aware evaluation, while also reporting diagnostic metrics that separate prediction quality from task-level utility. For variants without a latent world model or reward head, the corresponding diagnostic entries are omitted.

Table 3: **Ablation study.** Next-latent MSE is reported in 10^{-3} scale. Reward-Env Corr means reward-environment correlation.

Configuration	$\mathrm{SR}_{\mathrm{ID}} \uparrow$	$\mathrm{SR}_{\mathrm{OOD}} \uparrow$	Avg. SR $\uparrow$	Action $R^2 \uparrow$	Next-Latent MSE $\downarrow$	Reward-Env Corr. $\uparrow$
Direct Policy	0.20	0.08	0.14	0.23	–	–
Latent bottleneck only	0.25	0.12	0.19	0.28	–	–
World model, no reward head	0.23	0.10	0.17	0.25	**1.2**	–
Latent + world model, no refinement	0.38	0.24	0.31	0.32	2.3	0.32
Full LaRA-WM	**0.52**	**0.37**	**0.45**	**0.35**	2.7	**0.46**

The results in Table 3 reveal a useful distinction between representation quality and control usefulness. Adding only a latent bottleneck improves the direct policy modestly, increasing OOD success from 0.08 to 0.12, which suggests that compact action representations help but do not solve distribution shift. The world-model-only variant achieves the lowest next-latent MSE, yet its rollout performance remains close to the direct and latent-only baselines. This shows that accurate feature-space prediction can still be misaligned with task completion: a model may predict how features evolve without knowing which latent actions are actually useful for success.

The gap between the no-refinement variant and the full model highlights the role of test-time optimization. Without refinement, the model already benefits from latent action modeling and reward-aware representation learning, reaching 0.24 OOD success. The full LaRA-WM improves OOD success to 0.37, indicating that CEM-based refinement provides an additional execution-time correction mechanism. Although the full model does not minimize next-latent MSE, it achieves the

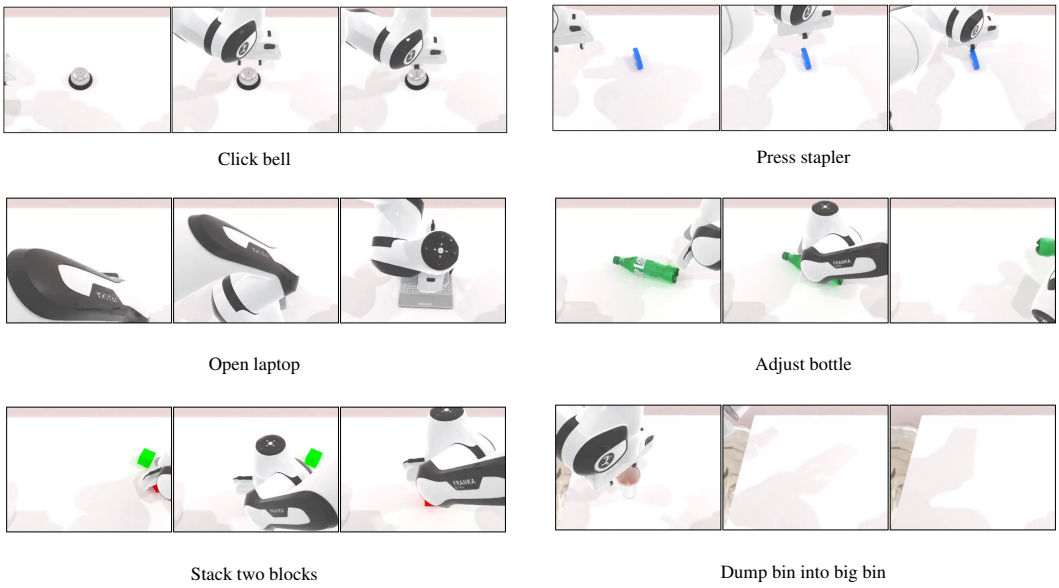

Figure 2: **OOD case studies across representative RoboTwin tasks.**These examples illustrate how test-time latent refinement helps correct execution-critical spatial and contact errors before actions are decoded into low-level motor commands.

highest reward-environment correlation, suggesting that the learned latent space is better aligned with task outcomes. Overall, the ablation supports the view that LaRA-WM works because prediction, reward alignment, and test-time search are combined into a single decision process.

4.4 Case Study

Figure 2 shows six representative OOD rollouts across diverse RoboTwin tasks, including *Click bell*, *Press stapler*, *Open laptop*, *Adjust bottle*, *Stack two blocks*, and *Dump bin into big bin*. These examples cover contact-triggering, articulated-object interaction, object pose adjustment, stacking, and transport behaviors. Across these tasks, the initial latent prediction typically captures the correct high-level intent but may still produce small spatial or contact misalignments that are difficult for reactive policies to recover from once the action is executed. LaRA-WM instead refines the latent action at test time using the learned reward scorer, leading to more task-consistent contact points, approach directions, and object interactions. As shown in Figure 2, this refinement is especially beneficial in OOD settings where small execution errors can cause complete failure. Overall, these qualitative results illustrate the main mechanism behind LaRA-WM: test-time latent refinement corrects execution-critical errors before latent intents are decoded into low-level motor commands.

5 Conclusion

We presented LaRA-WM, a latent action world model for robust robotic manipulation. Instead of directly mapping observations to low-level motor commands, LaRA-WM learns a compact task-centric latent action space and performs reward-guided test-time refinement before action execution. By using a learned reward-feature scorer and CEM-based latent optimization, the policy can evaluate and improve candidate action intents in a structured space rather than searching directly in the raw action space. Experiments on the RoboTwin benchmark show that LaRA-WM consistently improves both in-distribution and out-of-distribution task success rates over direct policies, latent-only variants, reward-free world-model variants, ACT, and Diffusion Policy. Ablation results further show that the main gains come from the combination of latent action modeling, reward-aware scoring, and iterative test-time refinement, rather than from latent compression alone. These results suggest that latent-space test-time optimization is an effective mechanism for improving robustness under visual and spatial distribution shifts. Future work will explore more efficient refinement strategies, better reward calibration, and longer-horizon latent planning for complex manipulation tasks.

AI USE STATEMENT

We used generative AI tools (large language model assistants) for two tasks in preparing this manuscript: (i) literature search and discovery of related work, and (ii) readability editing of the authors' own text, including sentence-level polishing, terminology consistency checks, and LaTeX formatting. We did not use generative AI tools to generate research hypotheses, design the method or the experiments, implement or debug code, produce proofs, process or analyze data, or interpret experimental results; the method, the implementation, and all reported numbers were produced by the authors. Every AI-suggested reference was verified by the authors against the original source, and all AI-assisted edits were reviewed line by line for technical and factual correctness. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 3 specifies the model, the training objectives, and the test-time refinement procedure, and Algorithm 1 gives the complete online refinement loop. Section 3.4 defines the evaluation protocol, the in-distribution and out-of-distribution splits, and the baselines. Appendix A lists the model components together with the implementation and training hyperparameters, Appendix B describes the ID and OOD settings, the observation and action preprocessing, and the diagnostic metrics, and Appendix C specifies every baseline, including its observation space, backbone, and training configuration. Appendix E reports the hardware and the approximate GPU-hours used for the reported experiments. All experiments are conducted in the publicly available RoboTwin simulation benchmark, so no proprietary data and no human-subject data are involved. Anonymous code, configuration files, and evaluation scripts will be provided as supplementary material to support reproduction of the reported results.

REFERENCES

Abbas Abdolmaleki, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Ashwin Balakrishna, Nathan Batchelor, Alex Bewley, Jeff Bingham, Michael Bloesch, Konstantinos Bousmalis, Philemon Brakel, Anthony Brohan, Thomas Buschmann, Arunkumar Byravan, Serkan Cabi, et al. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *arXiv preprint arXiv:2510.03342*, 2025.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

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

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 *9th Annual Conference on Robot Learning*, 2025. URL https://openreview.net/forum?id=vlhoswksBO.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023. *URL https://arxiv. org/abs/2307.15818*, 1:2, 2024.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2505.06111*, 2025.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Mingtong Dai, Lingbo Liu, Yongjie Bai, Yang Liu, Zhouxia Wang, Rui Su, Chunjie Chen, Liang Lin, and Xinyu Wu. Rover: Robot reward model as test-time verifier for vision-language-action model. *arXiv preprint arXiv:2510.10975*, 2025.

Zhekai Duan, Yuan Zhang, Shikai Geng, Gaowen Liu, Joschka Boedecker, and Chris Xiaoxuan Lu. Fast ecot: Efficient embodied chain-of-thought via thoughts reuse. *arXiv preprint arXiv:2506.07639*, 2025.

Shenyuan Gao, Siyuan Zhou, Yilun Du, Jun Zhang, and Chuang Gan. Adaworld: Learning adaptable world models with latent actions. *arXiv preprint arXiv:2503.18938*, 2025.

Shenyuan Gao, William Liang, Kaiyuan Zheng, Ayaan Malik, Seonghyeon Ye, Sihyun Yu, Wei-Cheng Tseng, Yuzhu Dong, Kaichun Mo, Chen-Hsuan Lin, et al. Dreamdojo: A generalist robot world model from large-scale human videos. *arXiv preprint arXiv:2602.06949*, 2026.

Yuhang Huang, Jiazhao Zhang, Shilong Zou, Xinwang Liu, Ruizhen Hu, and Kai Xu. Ladi-wm: A latent diffusion-based world model for predictive manipulation. *arXiv preprint arXiv:2505.11528*, 2025.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model, 2024. *URL https://arxiv. org/abs/2406.09246*, 1(2):4, 2024.

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

Seungjae Lee, Yibin Wang, Haritheja Etukuru, H. Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024.

Tony Lee, Andrew Wagenmaker, Karl Pertsch, Percy Liang, Sergey Levine, and Chelsea Finn. Roboreward: General-purpose vision-language reward models for robotics. *arXiv preprint arXiv:2601.00675*, 2026.

Yicheng Liu, Shiduo Zhang, Zibin Dong, Baijun Ye, Tianyuan Yuan, Xiaopeng Yu, Linqi Yin, Chenhao Lu, Junhao Shi, Luca Jiang-Tao Yu, Liangtao Zheng, Tao Jiang, Jingjing Gong, Xipeng Qiu, and Hang Zhao. Faster: Toward efficient autoregressive vision language action modeling via neural action tokenization. *arXiv preprint arXiv:2512.04952*, 2025.

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

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, and Xuelong Li. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Juan Rocamonde, Victoriano Montesinos, Elvis Nava, Ethan Perez, and David Lindner. Vision-language models are zero-shot reward models for reinforcement learning. In *International Conference on Learning Representations*, 2024.

Yu Shang, Lei Jin, Yiding Ma, Xin Zhang, Chen Gao, Wei Wu, and Yong Li. Longscape: Advancing long-horizon embodied world models with context-aware moe. *arXiv preprint arXiv:2509.21790*, 2025a.

Yu Shang, Xin Zhang, Yinzhou Tang, Lei Jin, Chen Gao, Wei Wu, and Yong Li. RoboScape: Physics-informed embodied world model. In *Advances in Neural Information Processing Systems*, 2025b.

Yinzhou Tang, Yu Shang, Yinuo Chen, Bingwen Wei, Xin Zhang, Shu'ang Yu, Liangzhi Shi, Chao Yu, Chen Gao, Wei Wu, et al. Roboscape-r: Unified reward-observation world models for generalizable robotics training via rl. *arXiv preprint arXiv:2512.03556*, 2025.

Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, Steven Bohez, Konstantinos Bousmalis, Anthony Brohan, Thomas Buschmann, Arunkumar Byravan, Serkan Cabi, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. Dexvla: Vision-language model with plug-in diffusion expert for general robot control. *arXiv preprint arXiv:2502.05855*, 2025.

Ze Yang, Yun Chen, Jingkang Wang, Sivabalan Manivasagam, Wei-Chiu Ma, Anqi Joyce Yang, and Raquel Urtasun. Unisim: A neural closed-loop sensor simulator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1389–1399, 2023.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, et al. Latent action pretraining from videos. *arXiv preprint arXiv:2410.11758*, 2024.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi Fan, and Joel Jang. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 3157–3181. PMLR, 2025.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Siyuan Zhou, Yilun Du, Jiaben Chen, Yandong Li, Dit-Yan Yeung, and Chuang Gan. Robodreamer: Learning compositional world models for robot imagination. *arXiv preprint arXiv:2404.12377*, 2024.

A Additional Method Details

This appendix provides implementation details, evaluation protocol clarifications, baseline descriptions, diagnostic metrics, compute notes, and limitations for LaRA-WM. We keep the appendix concise and primarily text-based. Quantitative summaries that are easier to interpret visually are reported in the main paper or in the corresponding result figures, while this appendix focuses on clarifying implementation and evaluation details.

A.1 Model Components

LaRA-WM is implemented as a latent action model for visual robotic manipulation. The model contains a backbone feature extractor, a latent encoder, a feature-space world model, an action decoder, and a latent refinement module. The key design choice is to avoid directly treating raw motor commands as the only decision variable. Instead, the model learns an intermediate latent representation that captures task-relevant action intent and then decodes this representation into low-level robot actions.

The trainable components are the latent encoder, the world-model module, the action decoder, and the refinement module. The latent encoder learns a compact action-oriented bottleneck. The world-model component imposes temporal structure on the latent representation by predicting action-conditioned feature transitions. The action decoder maps the latent representation to the robot action space. The refinement module is used to encourage the latent representation to be more task-aligned and predictive of useful actions.

The implementation uses two related representation spaces. The first is a compact latent action space. In the VAE-style latent encoder, this bottleneck has dimension 32. The second is a 1536-dimensional feature-space representation used by the world-model and decoder path. The compact latent is used to regularize action intent, while the 1536-dimensional feature-space representation preserves visual information needed for transition modeling and action decoding.

A.2 World Model and Refinement

The world-model module predicts action-conditioned transitions in feature space. It is not a pixel-level video prediction model. Its role is to provide a compact predictive structure over features and latent actions. This design keeps the method computationally lightweight while still encouraging the latent representation to encode action-relevant dynamics.

The refinement module uses this predictive structure to improve latent consistency. During training, the model can compare predicted latent transitions with target latent representations from subsequent observations. This provides auxiliary supervision beyond one-step action regression.

At deployment time, the intended refinement mechanism searches or adjusts latent action candidates before decoding. The main paper describes this as test-time latent refinement guided by a learned reward-feature scorer. In this appendix, we do not include a separate CEM hyperparameter table; the refinement mechanism is described conceptually here and implemented through the corresponding model and evaluation scripts.

A.3 Implementation Hyperparameters

We include only the core settings needed to interpret the implementation. Detailed architecture-specific settings are kept in the released configuration files.

B Evaluation Protocol

B.1 ID and OOD Settings

We evaluate model generalization under both in-distribution (ID) and out-of-distribution (OOD) settings. The ID setting uses the same object instances, scene layouts, lighting conditions, and initialization distributions as the training demonstrations. It therefore mainly measures the model's

Table 4: Core implementation and training hyperparameters.

Item	Value
Action dimension	7
Compact latent dimension	32
Feature-space dimension	1536
Latent encoder hidden dimension	256
World-model hidden dimension	512
Decoder hidden dimension	512
Dropout	0.1
Optimizer	AdamW
Batch size	8
Epochs	100
Learning rate	1×10^{-4}
Weight decay	1×10^{-4}
Gradient clipping norm	1.0
Warmup steps	1000
Scheduler	Cosine decay
Minimum learning rate	1×10^{-6}
Checkpoint frequency	Every 10 epochs
Best checkpoint metric	Validation total loss

ability to imitate and execute actions within the training distribution. In contrast, the OOD setting introduces controlled perturbations to evaluate robustness under changes in visual appearance, spatial initialization, and scene structure.

Specifically, the OOD setting includes four types of distribution shifts. The first is object-instance shift: ID evaluation uses object instances observed during training, while OOD evaluation uses held-out unseen instances from the same object category. The second is initial-pose shift: in the ID setting, object initial states are sampled from the pose range covered by demonstrations, whereas the OOD setting expands the translation range and yaw-angle range. The third is lighting shift: ID evaluation uses the default RoboTwin lighting, while OOD evaluation randomizes brightness, color temperature, and shadows. The fourth is scene-layout shift: ID evaluation uses backgrounds and layouts observed during training, while OOD evaluation uses held-out backgrounds, tabletop layouts, and distractor configurations.

Unless otherwise stated, the main OOD results reported in the paper use the combined-OOD setting. In this setting, the four types of perturbations may occur simultaneously rather than independently. Combined-OOD is therefore closer to the compound distribution shifts that may arise in real deployment, and better reflects the model's overall robustness to unseen objects, altered initialization, lighting changes, and novel scene layouts.

B.2 Observation and Action Preprocessing

All methods use the same observation preprocessing and action normalization pipeline. RGB images are resized to 224×224 and then passed into the frozen visual encoder to extract visual features. Language instructions are encoded using a fixed task-instruction template and remain unchanged within each episode of the same task.

Actions are normalized using the mean and standard deviation computed from the training split:

$$\tilde{a}_t = \frac{a_t - \mu_a}{\sigma_a + \epsilon}, \qquad \epsilon = 10^{-6}. \tag{12}$$

The model predicts normalized actions $\tilde{a}_t$. Before execution or error computation, predicted actions are clipped to $[-1, 1]$ in the normalized action space and then denormalized back to the original action space. All baselines and LaRA-WM use the same action normalization statistics to ensure a fair comparison.

C BASELINES

All methods use the same train, validation, and test splits, the same RoboTwin task list, the same maximum episode horizon, and the same validation-based hyperparameter selection protocol. Apart from the architectural differences between methods, observation preprocessing, action normalization, evaluation protocol, and success predicates are kept identical.

C.1 INTERNAL BASELINES

Direct Policy. Direct Policy is a reactive behavior-cloning baseline. It directly maps the current visual feature and language goal to a low-level action:

$$\hat{a}_t = \pi_{\text{direct}}(f_t, e_g). \tag{13}$$

This method uses the same frozen encoders and action normalization pipeline as LaRA-WM, but it does not use latent actions, a world model, a reward scorer, or CEM refinement. It therefore measures the baseline performance of a purely reactive observation-to-action policy under both ID and OOD settings.

Latent No-Refine. Latent No-Refine uses the same latent action representation and action decoder as LaRA-WM, but disables CEM refinement at test time. It directly decodes the initial latent prediction:

$$\hat{a}_t = D_\theta(f_t, \hat{z}_t, e_g). \tag{14}$$

This baseline isolates the effect of the latent action bottleneck itself, measuring how much benefit is obtained from using a latent action representation without test-time refinement.

No-Reward WM. No-Reward WM keeps the latent action encoder and the feature-space transition head, but removes the reward scorer. Since it does not have a reward-aware score for CEM, it does not perform reward-guided latent refinement. This baseline tests whether learning feature-space transitions alone is sufficient to improve control performance, or whether reward-aware candidate scoring is necessary for stronger OOD generalization.

C.2 EXTERNAL BASELINES

ACT. ACT is an action-chunking imitation-learning baseline. It predicts a chunk of future actions and executes them in a receding-horizon manner. We use frozen DINOv2 features as observations. The Transformer backbone has 4 layers, hidden dimension 512, 8 attention heads, and an action chunk size of 10. ACT is trained with AdamW using a learning rate of 1×10^{-4} for 100 epochs. This baseline represents a standard action-chunking imitation-learning approach.

Diffusion Policy. Diffusion Policy is trained as a visuomotor diffusion baseline. It uses an observation horizon of 2, a prediction horizon of 16, and an action horizon of 8. The model uses 100 diffusion steps during training and 16 denoising steps during inference, with a cosine noise schedule. As with other methods, Diffusion Policy uses frozen DINOv2 features as observations. It is trained with AdamW using a learning rate of 1×10^{-4} for 100 epochs. This baseline compares generative action modeling with latent refinement.

OpenVLA-oft. OpenVLA-oft is a fine-tuned vision-language-action baseline. We use LoRA adapters for parameter-efficient fine-tuning and add a 7-D continuous action regression head matched to the RoboTwin action interface. The LoRA rank is 32, LoRA alpha is 64, and LoRA dropout is 0.05. Fine-tuning uses a learning rate of 2×10^{-5}, batch size 8, and 20 epochs. Input images are resized to 224×224, and prompts use the same task-instruction template as LaRA-WM. This baseline compares LaRA-WM with a fine-tuned VLA policy under the same continuous action interface.

D METRICS AND DIAGNOSTICS

In addition to task success rate, we report several diagnostic metrics to analyze action prediction quality, latent transition modeling quality, and the alignment between the reward scorer and environment feedback.

D.1 ACTION PREDICTION METRIC

Action R^2. Action R^2 measures how much action variance is explained by the model:

$$R^2 = 1 - \frac{\sum_t \|a_t - \hat{a}_t\|_2^2}{\sum_t \|a_t - \bar{a}\|_2^2}. \tag{15}$$

Here, a_t is the ground-truth demonstration action, $\hat{a}_t$ is the predicted action, and $\bar{a}$ is the mean action in the test set. A higher R^2 indicates that the model explains more of the variation in the demonstrated actions.

D.2 LATENT TRANSITION METRIC

Next-Latent MSE. Next-Latent MSE measures feature-space transition prediction error:

$$\mathrm{MSE}_{\mathrm{next}} = \frac{1}{T} \sum_t \|\hat{f}_{t+1} - f_{t+1}\|_2^2. \tag{16}$$

Here, $\hat{f}_{t+1}$ is the next-step feature predicted by the world model, and f_{t+1} is the target feature encoded from the next observation. This metric evaluates the short-horizon feature prediction quality of the world model and is reported in the ablation table at a scale of 10^{-3}.

D.3 REWARD-ALIGNMENT METRIC

Reward-Env Corr. Reward-environment correlation measures the alignment between the reward scorer and environment feedback. Specifically, we compute the Pearson correlation between predicted reward scores and environment rewards or success labels:

$$\rho_{\mathrm{reward}} = \mathrm{corr}\left(S_\psi(f_t, z_t, e_g), r_t^{\mathrm{env}}\right). \tag{17}$$

A higher correlation indicates that the scorer better distinguishes latent actions that are useful for task success, making it more suitable for CEM-based candidate selection.

D.4 REFINEMENT METRIC

Refinement Gain. Refinement gain measures the improvement brought by test-time latent refinement. It is defined as the difference in success rate between LaRA-WM and the latent baseline without refinement:

$$\mathrm{RG} = \mathrm{SR}_{\text{LaRA-WM}} - \mathrm{SR}_{\text{Latent No-Refine}}. \tag{18}$$

This metric directly quantifies how much CEM-based latent refinement improves execution beyond using a latent action representation alone.

E COMPUTE AND RESOURCE

All experiments were implemented in PyTorch and executed with CUDA acceleration. The main experiments were run on a workstation equipped with NVIDIA H200 GPUs; training and evaluation used cached or pretrained backbone features whenever available. We distinguish offline training compute from native simulator rollout compute: the former is performed on a fixed demonstration dataset and is relatively stable, while the latter is affected by RoboTwin simulator stepping, environment reset overhead, task horizon, early termination, and hardware load, so wall-clock time may vary noticeably across tasks and machines.

For LaRA-WM, one training run required approximately 6.4 GPU-hours on one H200 GPU. Including three random seeds and validation sweeps, the full experiment matrix required approximately

165 GPU-hours; this estimate excludes exploratory debugging runs, failed infrastructure or simulator runs, and non-final hyperparameter probes that were not used for model selection. At test time, LaRA-WM uses 5 CEM iterations with 64 latent candidates per iteration. Candidate scoring is batched on GPU, resulting in a policy latency of approximately 52 ms per control step, which satisfies the 10 Hz closed-loop control budget used in evaluation. Native simulator evaluation used 50 rollouts per task and seed, with three seeds per method. Since rollout cost depends strongly on task horizon and simulator behavior, simulator wall-clock time is reported separately from offline training compute and is not included in the GPU-hour total.

F Discussion and Limitation

The results suggest that LaRA-WM improves manipulation performance mainly by enabling better test-time action selection, rather than merely improving one-step imitation accuracy. By refining candidate actions in a compact latent space with reward-aware scoring, the model can adjust its action intent before execution and achieve stronger OOD generalization. At the same time, several aspects remain worth further exploration. First, offline action prediction is only a partial proxy for closed-loop manipulation, since it does not fully capture recovery, contact timing, or long-horizon error accumulation. Second, the reward scorer plays an important role during refinement, so future work could study better calibration and uncertainty-aware scoring to avoid over-trusting imperfect reward estimates. Finally, while our evaluation focuses on simulation, real-robot deployment would require additional validation of latency, safety constraints, workspace limits, and hardware-specific calibration. We view these directions as natural extensions for making latent-space test-time refinement more reliable in broader robotic settings.

G Broader Impact, Safety, and Asset Usage

G.1 Broader impact and safety

LaRA-WM aims to improve the robustness of robotic manipulation policies under distribution shift. More reliable learned policies may reduce the need for manual intervention in structured manipulation environments and improve the stability of deployed robotic systems. At the same time, the primary safety risk comes from physical execution errors. In particular, the learned reward scorer may mis-rank candidate latent actions, and test-time optimization may amplify such errors before the selected latent action is decoded and executed.

Potential mitigations include action clipping, workspace constraints, latent-prior regularization, uncertainty-aware reward scoring, simulator-first validation, task-level safety monitors, and human override mechanisms during real-world deployment. We view these safeguards as important requirements for transferring LaRA-WM from simulation to physical robotic systems.

G.2 Assets and licenses

Asset	Usage	Redistribution	License
RoboTwin	Benchmark and simulation data	Not redistributed	MIT
DINOv2	Frozen visual encoder	Not redistributed	Apache-2.0
CLIP	Frozen text encoder	Not redistributed	MIT
ACT	Baseline implementation	Not redistributed	MIT
Diffusion Policy	Baseline implementation	Not redistributed	MIT
OpenVLA	VLA baseline	Not redistributed	MIT

All third-party assets are used only for research evaluation. We do not redistribute RoboTwin assets, pretrained foundation-model weights, or baseline checkpoints. The licenses listed above refer to the corresponding public releases used in our experiments.

G.3 LLM AND FOUNDATION-MODEL USAGE

The core LaRA-WM method does not use a large language model as a planner, code generator, executor, evaluator, or reward annotator. Foundation models are used only for representation extraction or baseline comparison: DINOv2 serves as the frozen visual encoder, CLIP serves as the frozen text encoder, and OpenVLA-oft is included only as an external VLA baseline. No external closed-source API is used for action selection, reward computation, rollout evaluation, or success labeling in the reported experiments.