Never Stop Learning: Test-Time Reinforcement Learning for Vision-Language-Action Models

Xinying Yi **Jingjing Jiang** **Chao Ma***
MoE Key Lab of Artificial Intelligence, Institute of AI
Shanghai Jiao Tong University, Shanghai, China
xyyi0623@sjtu.edu.cn jingjingjiang2017@gmail.com
chaoma@sjtu.edu.cn

Project Page: https://swilder7.github.io/explore-vla

Abstract

Test-time compute scaling offers a promising path for improving foundation models beyond training-time data and parameters. For Vision-Language-Action (VLA) models, however, applying this paradigm through online reinforcement learning remains challenging due to two coupled limitations. Candidate chunks are often biased toward the pretrained policy's high-likelihood regions, while sparse outcome-level feedback lacks process grounding, making reward signals weakly discriminative and exploration insufficiently guided. To address these challenges, we propose **EXPLORE**, a simulator-based test-time reinforcement learning framework that coordinates **Explo**itation and **Explo**ration for action-chunk generation and drives policy optimization with physics-grounded process **re**wards. Through this joint design, EXPLORE broadens the effective action space while preserving the pretrained prior and derives faithful process rewards from dense physical interaction feedback to guide online policy improvement. Experiments on SimplerEnv-WidowX, DexMG, and RoboCasa validate the effectiveness of EXPLORE, with consistent improvements across autoregressive and diffusion-based VLA backbones.

1 Introduction

Scaling test-time compute [8, 29, 35, 61, 72, 76, 86] has recently emerged as a complementary dimension of capability scaling for large foundation models, beyond training-time data and parameters. This regime relies on tractable scoring signals over candidate outputs that can be evaluated without physical interaction with the environment, such as rule-based verifiers [11, 41] and learned reward models [15, 42]. However, such signals remain unavailable for Vision-Language-Action (VLA) models in robotic manipulation [7, 10, 19, 54, 57, 71, 87], with policies pretrained from demonstration data and verifiable feedback arising only from environment execution. This constrains test-time scaling for VLAs in two coupled ways tied to the pretrained policy distribution and the structure of execution feedback. Reinforcement learning (RL) [64] offers a natural paradigm of test-time scaling for VLAs through online adaptation from execution feedback [2, 44, 79], yet realizing it on leading foundation VLAs leaves these two coupled challenges without a joint solution.

In this paper, we focus on two coupled structural challenges. ***(i)*** **Action-distribution narrowness.** A pretrained VLA concentrates its actions within a narrow high-likelihood region shaped by demonstration data. Consequently, on-policy sampling produces near-duplicate candidates whose rewards offer no informative gradient signal for policy updates, while naively broadening the sampling distribution erodes the pretrained prior that test-time RL is intended to refine. ***(ii)*** **Reward sparsity and**

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

weak grounding. A complete task execution returns only a binary outcome at episode end. Such sparse outcomes (*e.g.*, the delayed feedback [79]) preclude per-step credit assignment over long-horizon trajectories. While learned process critics have been introduced to provide denser process rewards [2, 44], they tend to drift on out-of-distribution rollouts and yield rewards weakly calibrated to physical task progress. These two challenges are deeply coupled, as broadened exploration without process-grounded rewards reduces to random perturbation, whereas faithful rewards without diverse candidates collapse to uninformative gradients. Addressing them therefore requires a joint design that simultaneously expands the candidate distribution under controlled deviation from the pretrained prior and supplies dense, physics-grounded process supervision.

We propose **EXPLORE**, a simulator-based test-time reinforcement learning framework that addresses both challenges within a per-decision-step optimization loop. To resolve action-distribution narrowness, EXPLORE employs *coordinated exploration-exploitation action-chunk generation*, constructing at each decision step a candidate group that combines high-confidence action chunks drawn from the current policy with exploratory chunks from a deliberately broadened distribution. This design effectively expands the action space without abandoning the pretrained prior. To address reward sparsity and weak grounding, EXPLORE performs *physics-grounded process reward modeling*, rolling out each candidate from a shared cached simulator state and accumulating step-wise rewards from simulated physical feedback into a comparable, progress-aligned chunk-level score. Subsequently, EXPLORE restores the simulator to the cached state, appends the highest-scoring action chunk to the executed trajectory, and updates the policy from the reward-scored candidate group via an architecture-adaptive GRPO applicable to both autoregressive and diffusion-based VLA architectures.

We evaluate EXPLORE on three benchmarks, including SimplerEnv-WidowX [39], DexMG [30], and RoboCasa [51], and observe consistent improvements across two typical VLA architectures, namely the autoregressive-based UniVLA [69] and the diffusion-based ACG [55]. Notably, EXPLORE raises UniVLA on SimplerEnv-WidowX from 69.8% to 81.3% (**+11.5** absolute) and boosts ACG on DexMG from 36.27% to 46.19% (**+9.92**), demonstrating the effectiveness of the proposed framework and establishing the practical value of test-time RL for VLA models.

Our contributions are summarized as follows:

- We propose EXPLORE, a simulator-based test-time reinforcement learning framework that performs per-decision-step policy improvement from dense interaction feedback, enabling VLA models to adapt to complex robotic manipulation.
- We develop *coordinated exploration-exploitation action-chunk generation*, which constructs complementary exploitation and exploration candidates to preserve the pretrained policy prior while broadening the effective action space.
- We introduce *physics-grounded process reward modeling*, which evaluates candidate action chunks via dense feedback from simulator interaction, providing faithful process-level rewards for VLA test-time reinforcement learning.

2 Related Work

Vision-Language-Action Models. VLA models can be broadly organized by their training and adaptation stages. *Vanilla VLA models* learn general-purpose robot policies from broad robot, web, video, or multimodal data, spanning large-scale robot transformers [7, 54, 87], open generalist policies [33, 38, 53], spatial and world-model-based policies [10, 19, 37, 57, 71, 83], and diffusion-based action models [4, 5, 23, 45, 46, 60, 70, 75]. *Post-trained VLA models* further specialize these policies through task-specific fine-tuning [34, 43, 84], world-dynamics post-training [26, 32, 69], spatial robustness training [12, 78, 82], temporal memory modeling [25, 59, 67], embodiment-aware adaptation [9, 18], and reinforcement-learning alignment [13, 14, 22, 36, 47, 77, 80]. These training-stage advances have greatly improved VLA policies before deployment, while a complementary family of *test-time adapted VLA models* seeks to further improve inference through additional computation or online feedback. Existing methods either allocate additional test-time computation without updating the base policy [27, 35, 50, 55, 73], or adapt the policy from test-time feedback [2, 44, 79]. For example, EVOLVE-VLA [2] uses a learned progress estimator for environment-feedback adaptation, TT-VLA [44] performs on-the-fly adaptation with step-wise task-progress rewards, and PDF [79] learns a lightweight perturbation module from delayed feedback. EXPLORE instead performs architecture-adaptive test-time reinforcement learning over VLA action chunks. It coordinates

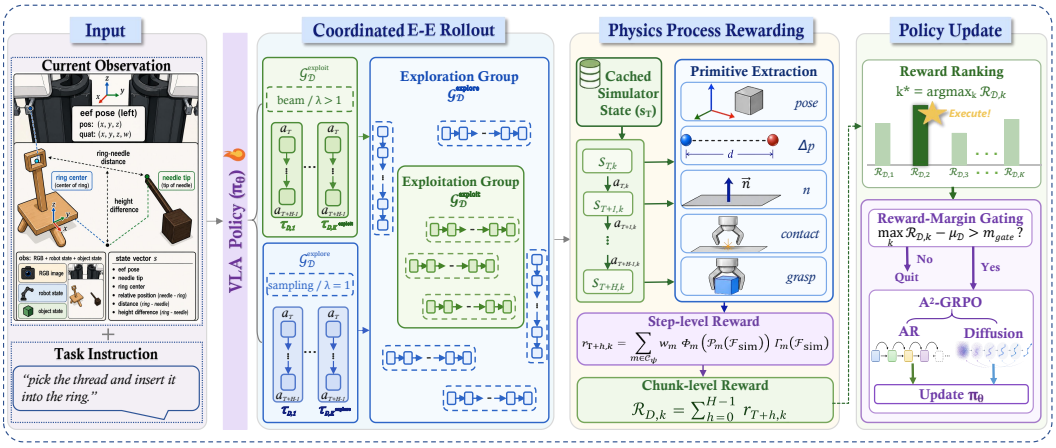


Figure 1: **Method overview.** EXPLORE is a test-time reinforcement learning framework for VLA models. It coordinates exploitation and exploration to generate diverse candidate action chunks, evaluates them with physics-grounded process rewards, executes the highest-scoring chunk, and updates the policy online with an architecture-adaptive GRPO (A^2-GRPO).

exploitation and exploration to construct reward-discriminative candidate groups, evaluates them with physics-grounded process rewards, and turns process-level feedback into online policy updates.

Test-Time Learning. Test-time learning can be broadly divided into test-time scaling without parameter updates and test-time training with online updates. Test-time scaling improves inference by allocating extra computation to sampling and search [61, 68, 74, 85], reward-guided decoding or verification [17, 81], and parameter-free prediction adjustment under distribution shift [6, 21, 24, 31, 40]. In contrast, test-time training adapts model parameters or lightweight modules during inference. For example, TTT [62] optimizes a self-supervised objective on unlabeled test samples, while Tent [65] performs online adaptation by minimizing prediction entropy. This paradigm has also been extended to vision-language and reasoning settings [1, 3, 16, 20, 49, 52, 63, 66]. For VLA models, test-time learning has only recently been explored through environment-feedback adaptation and perturbation learning [2, 44, 79]. Different from methods that mainly adjust predictions, prompts, representations, or auxiliary modules, EXPLORE optimizes action-level behavior through coordinated candidate action chunks. Its update signal comes from physics-grounded process rewards, rather than prediction confidence or self-supervised consistency alone.

3 Methodology

Problem Formulation. Robotic manipulation under language instructions is commonly formulated as a partially observable Markov decision process (POMDP),

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{L}), \tag{1}$$

where $\mathcal{S}$ denotes the latent state space of the environment, $\mathcal{A}$ is the action space, $\mathcal{O}$ is the observation space, and $\mathcal{L}$ represents the space of task instructions. Given an instruction $l \in \mathcal{L}$, **at the $\mathcal{D}$-th decision step**, the VLA policy $\pi_{\boldsymbol{\theta}}$ receives a corresponding observation $o_{\mathcal{D}} \in \mathcal{O}$ and generates a short-horizon action chunk $\tau_{\mathcal{D}} = (a_T, a_{T+1}, \cdots, a_{T+H-1})$,

$$\tau_{\mathcal{D}} \sim \pi_{\boldsymbol{\theta}}(\cdot \mid o_{\mathcal{D}}, l), \tag{2}$$

where each decision step starts at the action index $T = \mathcal{D}H$, with H denoting the chunk horizon.

Method Overview. As illustrated in Figure 1, we propose a test-time reinforcement learning framework (EXPLORE), enabling VLA models to self-improve continuously at test time. At each decision step, the proposed EXPLORE first coordinates exploitation and exploration to generate a diverse candidate group of action chunks (Sec. 3.1), which are then evaluated with a physics-grounded process reward that converts dense environment feedback into chunk-level scores (Sec. 3.2), and finally integrates an architecture-adaptive group relative policy optimization (A^2-GRPO) to turn the group rewards into an in-place policy update (Sec. 3.3).

3.1 Coordinated Exploration-Exploitation Action-Chunk Generation

To mitigate test-time distribution shift, we coordinate exploration and exploitation to broaden the coverage of the effective action space beyond baseline VLAs' high-confidence inference distribution. Specifically, at each decision step $\mathcal{D}$, we construct two complementary groups of candidate actions: an exploitation group $\mathcal{G}_{\mathcal{D}}^{\text{exploit}}$ and an exploration group $\mathcal{G}_{\mathcal{D}}^{\text{explore}}$,

$$\begin{aligned} \mathcal{G}_{\mathcal{D}}^{\text{exploit}} &= \left\{ \tau_{\mathcal{D},i} \right\}_{i=1}^{K^{\text{exploit}}}, \quad \tau_{\mathcal{D},i} \sim \pi_{\boldsymbol{\theta}}(\cdot \mid o_{\mathcal{D}}, l), \\ \mathcal{G}_{\mathcal{D}}^{\text{explore}} &= \left\{ \tau_{\mathcal{D},j} \right\}_{j=1}^{K^{\text{explore}}}, \quad \tau_{\mathcal{D},j} \sim \tilde{\pi}_{\boldsymbol{\theta}}(\cdot \mid o_{\mathcal{D}}, l), \end{aligned} \tag{3}$$

where $\pi_{\boldsymbol{\theta}}$ denotes the high-confidence action distribution of the baseline VLA, and $\tilde{\pi}_{\boldsymbol{\theta}}$ denotes its broadened variant covering the full action distribution (*e.g.*, via temperature scaling). Let $T := \mathcal{D}H$ denote the starting action index, where H is the horizon of an action chunk. The current policy $\pi_{\boldsymbol{\theta}}$ constructs a candidate group $\mathcal{G}_{\mathcal{D}} = \{\tau_{\mathcal{D},k}\}_{k=1}^{K}$ conditioned on the corresponding observation $o_{\mathcal{D}}$ and instruction l, where $K = K^{\text{exploit}} + K^{\text{explore}}$ with $K^{\text{exploit}}, K^{\text{explore}} \in \mathbb{N}_{>0}$. Each candidate $\tau_{\mathcal{D},k}$ is generated by $\pi_{\boldsymbol{\theta}}$ under a control variable ξ_k, whose admissible space depends on the action-head architecture:

$$\tau_{\mathcal{D},k} = \text{Gen}(\pi_{\boldsymbol{\theta}}, o_{\mathcal{D}}, l; \xi_k), \quad \xi_k \in \begin{cases} \Xi^{\text{exploit}}, & \text{if } k \le K^{\text{exploit}}, \\ \Xi^{\text{explore}}, & \text{otherwise}, \end{cases} \tag{4}$$

and the resulting $\tau_{\mathcal{D},k}$ is assigned to $\mathcal{G}_{\mathcal{D}}^{\text{exploit}}$ or $\mathcal{G}_{\mathcal{D}}^{\text{explore}}$ accordingly. For autoregressive VLA heads, sampling ξ_k from Ξ^{exploit} corresponds to beam search or beam sampling that favors *high-likelihood action chunks*, while sampling from Ξ^{explore} corresponds to ancestral sampling that yields *diverse rollouts*. Analogously, for diffusion-based action heads, ξ_k is instantiated as the classifier-free guidance scale λ_k, with $\Xi^{\text{exploit}} = \{\lambda_k : \lambda_k > 1\}$ and $\Xi^{\text{explore}} = \{1\}$.

These candidate action chunks $\{\tau_{\mathcal{D},k}\}_{k=1}^{K}$ are then scored by the proposed physics-grounded process reward in Eq. (7). We select the highest-scoring action chunk $\tau_{\mathcal{D}}^{\star}$ for execution over $[T, T+H-1]$, and reuse the full candidate group with its rewards as process-level supervision for the subsequent policy update. In summary, the coordinated exploitation-exploration design preserves the pretrained policy prior while broadening the coverage and enhancing the diversity of the action space.

3.2 Physics-Grounded Process Reward Modeling

To address the sparsity of outcome rewards (*e.g.*, success or failure) and the limited physical grounding of learned reward models, we propose *physics-grounded process reward modeling*, which leverages dense interaction feedback, including object poses, contact relations, grasp states, and task-relevant kinematic constraints, obtained by rolling out candidate trajectories in a physics simulator. The resulting reward preserves meaningful variation across candidate action chunks, yielding more faithful feedback on robotic manipulation progress for group-based policy optimization in Sec. 3.3.

Process Reward Formulation. We introduce a *state-caching and environment-restoration mechanism* within the simulator to ensure that all candidate action chunks are evaluated from the same initial state. Formally, for the decision step $\mathcal{D}$ with starting action index $T = \mathcal{D}H$, we evaluate each candidate action chunk $\tau_{\mathcal{D},k} = (a_{T,k}, a_{T+1,k}, \cdots, a_{T+H-1,k})$ by rolling it out from the cached simulator state s_T. For candidate-indexed notation, we write this shared initial state as $s_{T,k}$ for the k-th rollout. The subsequent candidate-specific states evolve as follows:

$$s_{T+h+1,k} = \mathcal{F}_{\text{sim}}(s_{T+h,k}, a_{T+h,k}), \quad h = 0, 1, \ldots, H-1, \tag{5}$$

where $\mathcal{F}_{\text{sim}}$ is the simulator transition function, and $s_{T+h,k}$ denotes the simulator state of the k-th candidate at the action step $T+h$. Instead of learning a separate critic, we define each step reward r_{ψ} analytically as a weighted composition of physics-aware physical primitives:

$$r_{\psi}(s_{T+h,k}, a_{T+h,k}) = \sum_{m \in \mathcal{C}_{\psi}} w_m \, \Phi_m\Big(\mathcal{P}_m(\mathcal{F}_{\text{sim}}(s_{T+h,k}, a_{T+h,k}))\Big) \, \Gamma_m(\mathcal{F}_{\text{sim}}(s_{T+h,k}, a_{T+h,k})), \tag{6}$$

where $\mathcal{C}_{\psi}$ denotes the set of active reward terms for task configuration ψ, m indexes one such term, and w_m is its scalar weight. $\mathcal{P}_m$ extracts physics-aware observables from the post-transition simulator state $s_{T+h+1,k}$, Φ_m maps them to smooth reward values, and $\Gamma_m \in \{0, 1\}$ activates the

Algorithm 1 A^2-GRPO Update at Decision Step $\mathcal{D}$

Require: Policy $\pi_{\boldsymbol{\theta}_\mathcal{D}}$, action chunks $\mathcal{G}_\mathcal{D} = \{\tau_{\mathcal{D},k}\}_{k=1}^K$, rewards $\{\mathcal{R}_{\mathcal{D},k}\}_{k=1}^K$, hyperparameters m_{gate}

$\mu_\mathcal{D} \leftarrow \frac{1}{K}\sum_{k=1}^K \mathcal{R}_{\mathcal{D},k}, \quad \sigma_\mathcal{D} \leftarrow \sqrt{\frac{1}{K}\sum_{k=1}^K (\mathcal{R}_{\mathcal{D},k} - \mu_\mathcal{D})^2}$

$k^\star \leftarrow \arg\max_k \mathcal{R}_{\mathcal{D},k}, \quad \tau_\mathcal{D}^\star \leftarrow \tau_{\mathcal{D},k^\star}$

if $\mathcal{R}_{\mathcal{D},k^\star} - \mu_\mathcal{D} \le m_{\text{gate}}$ **then** ▷ Reward-margin gating using Eq. (9)

  $\boldsymbol{\theta}_{\mathcal{D}+1} \leftarrow \boldsymbol{\theta}_\mathcal{D}$

else

  $\hat{A}_{\mathcal{D},k} \leftarrow (\mathcal{R}_{\mathcal{D},k} - \mu_\mathcal{D})/(\sigma_\mathcal{D} + \epsilon_{\text{adv}}),\ k = 1, 2, \ldots, K$

  Compute $\{\ell_{\mathcal{D},k}^{\text{head}}(\boldsymbol{\theta})\}_{k=1}^K$ for $\{\tau_{\mathcal{D},k}\}_{k=1}^K$ ▷ Instantiate head term with Eq. (11)

  Set head regularizer $\Omega_\mathcal{D}^{\text{head}}$

  $\mathcal{L}_{\text{A}^2\text{-GRPO},\mathcal{D}} \leftarrow \frac{1}{K}\sum_{k=1}^K \ell_{\mathcal{D},k}^{\text{head}}(\boldsymbol{\theta}) + \Omega_\mathcal{D}^{\text{head}}$

  $\boldsymbol{\theta}_{\mathcal{D}+1} \leftarrow \boldsymbol{\theta}_\mathcal{D} - \eta \cdot \text{ClipGrad}\left(\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{A}^2\text{-GRPO},\mathcal{D}},\ \delta_{\max}\right)$

end if

return $\boldsymbol{\theta}_{\mathcal{D}+1},\ \tau_\mathcal{D}^\star$

term only when its physical preconditions are satisfied. Under task-specific reward configuration ψ, the chunk-level process reward accumulates step-wise rewards over the short-horizon (H) rollout:

$$\mathcal{R}_{\mathcal{D},k} = \sum_{h=0}^{H-1} r_\psi(s_{T+h,k}, a_{T+h,k}), \quad k = 1, 2, \ldots, K. \tag{7}$$

Physical Interaction Primitives. The observables extracted by $\mathcal{P}_m$ summarize task-relevant physical states and interaction patterns, including object pose, contact relations, grasp status, and relative spatial relations. In practice, Φ_m is instantiated with a compact set of smooth potential functions, including exponential distance decay, sigmoid proximity bonuses, and axis-projected progress terms:

$$\Phi_m \in \left\{\exp(-\alpha\|\Delta\mathbf{p}\|),\ \sigma(-\alpha d),\ \max(0, \Delta\mathbf{p}^\top \mathbf{n})\right\}, \tag{8}$$

where $\Delta\mathbf{p}$ denotes relative displacement, d denotes signed geometric distance, $\mathbf{n}$ is a task-specific normal or motion axis, α is a positive scaling coefficient, and $\sigma(\cdot)$ denotes the sigmoid function.

3.3 Test-Time Policy Optimization with Architecture-Adaptive GRPO

As summarized in Algo. 1, EXPLORE performs test-time adaptation through an iterative per-decision-step optimization loop. At the $\mathcal{D}$-th decision step, it takes the candidate action chunks $\{\tau_{\mathcal{D},k}\}_{k=1}^K$ obtained in Sec. 3.1 and the rewards $\{\mathcal{R}_{\mathcal{D},k}\}_{k=1}^K$ yielded by the physics-grounded process reward in Sec. 3.2, and updates the current policy $\pi_{\boldsymbol{\theta}_\mathcal{D}}$ online via an architecture-adaptive GRPO. The algorithm finally returns both the updated parameters $\boldsymbol{\theta}_{\mathcal{D}+1}$ and the selected chunk $\tau_\mathcal{D}^\star$. We next detail test-time policy optimization with our architecture-adaptive GRPO.

Reward-Margin Gating. *For memory-efficient adaptation under the expanded action space*, we strategically adopt lightweight GRPO-style updates in the native output space of each VLA action head. Given the physics-grounded rewards $\{\mathcal{R}_{\mathcal{D},k}\}_{k=1}^K$ within a group, a critic-free, group-relative advantage $\hat{A}_{\mathcal{D},k}$ is obtained by normalizing each reward against the group mean $\mu_\mathcal{D}$ and standard deviation $\sigma_\mathcal{D}$ [58]. However, such normalization is scale-invariant. When all candidates in a group attain similarly low rewards, $\hat{A}_{\mathcal{D},k}$ is still pushed toward ± 1, injecting noisy gradients that destabilize test-time optimization. We therefore introduce *Reward-Margin Gating*, which admits an update only when the best-in-group reward clears a margin m_{gate} above the mean:

$$g_\mathcal{D} \triangleq \mathbb{1}\left[\max_k \mathcal{R}_{\mathcal{D},k} - \mu_\mathcal{D} > m_{\text{gate}}\right]. \tag{9}$$

The policy is updated only when $g_\mathcal{D} = 1$ and silently held fixed otherwise.

Architecture-Adaptive Group Relative Policy Optimization (A^2-GRPO). The test-time group objective shares a common form across action decoding heads,

$$\mathcal{L}_{\text{A}^2\text{-GRPO},\mathcal{D}} = \frac{1}{K}\sum_{k=1}^K \ell_{\mathcal{D},k}^{\text{head}}(\boldsymbol{\theta}) + \Omega_\mathcal{D}^{\text{head}}, \tag{10}$$

Table 1: **Comparison of test-time adaptation strategies** across action selection criteria, VLA update (Upd), RL-based optimization (RL), average SR (%), and inference time (h).

Methods	Action Selection Criterion	Upd	RL	DexMG		SimplerEnv	
				Average SR	Time	Average SR	Time
Baseline	–	–	–	36.27	2.01	69.79	1.59
Best-of-N	physics-grounded score	✗	✗	42.44 (+ 6.17)	14.04	72.92 (+ 3.13)	4.43
LAME-style [6]	the conditional entropy of prediction	✗	✗	37.96 (+ 1.69)	32.74	66.67 (- 3.12)	6.93
Tent-style [65]	the probability entropy of prediction	✓	✗	37.96 (+ 1.69)	4.49	67.71 (- 2.08)	1.74
EXPLORE	**our physics-grounded process reward**	✓	✓	**46.19 (+ 9.92)**	16.84	**81.25 (+ 11.46)**	7.61

where $\ell^{\text{head}}_{\mathcal{D},k}$ is a head-native term that realizes the advantage-weighted policy update for the k-th action chunk, and $\Omega^{\text{head}}_{\mathcal{D}}$ is a head-specific regularizer. Specifically, $\ell^{\text{head}}_{\mathcal{D},k}$ is given by

$$\ell^{\text{head}}_{\mathcal{D},k}(\boldsymbol{\theta}) = \begin{cases} -\operatorname{sg}\Big(\hat{A}_{\mathcal{D},k}\Big) \sum_{u=1}^{L_k} \log \pi_{\boldsymbol{\theta}}\left(z_{k,u} \mid o_{\mathcal{D}}, l, z_{k,<u}\right), & \text{autoregressive head,} \\ -\min\Big(\rho_{\mathcal{D},k}\hat{A}_{\mathcal{D},k},\ \bar{\rho}_{\mathcal{D},k}\hat{A}_{\mathcal{D},k}\Big), & \text{diffusion head,} \end{cases} \tag{11}$$

where $z_{k,u}$ denotes the u-th generated action token of $\tau_{\mathcal{D},k}$ with sequence length L_k, and the loss-space ratio $\rho_{\mathcal{D},k}$ together with its clipped counterpart $\bar{\rho}_{\mathcal{D},k}$ will be specified below.

In the autoregressive case, the log-likelihood is summed only over the valid action-token vocabulary. We set $\Omega^{\text{head}}_{\mathcal{D}} = 0$, since each decision step performs a single on-policy gradient step that obviates importance-ratio clipping. *In the diffusion case*, we instead construct a GRPO-clipped surrogate in loss space due to the lack of an explicit per-token likelihood. We replay the cached sampling path from the stored initial noise and conditioning features, recompute the native velocity-matching loss $\mathcal{L}_{\text{vel},\mathcal{D},k}(\boldsymbol{\theta})$, and define the loss-space ratio as

$$\rho_{\mathcal{D},k} = \exp(-\mathcal{L}_{\text{vel},\mathcal{D},k}(\boldsymbol{\theta}) + \operatorname{sg}(\mathcal{L}_{\text{vel},\mathcal{D},k}(\boldsymbol{\theta}))). \tag{12}$$

This surrogate satisfies $\rho_{\mathcal{D},k} \equiv 1$ in value while $\nabla_{\boldsymbol{\theta}}\rho_{\mathcal{D},k} = -\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{vel},\mathcal{D},k}$, mimicking the importance ratio in loss space. The clipped counterpart $\bar{\rho}_{\mathcal{D},k}$ restricts $\rho_{\mathcal{D},k}$ to the interval $[1-\epsilon, 1+\epsilon]$, and the regularizer is $\Omega^{\text{head}}_{\mathcal{D}} = -\beta\mathcal{H}_{\text{nat},\mathcal{D}}$, where $\mathcal{H}_{\text{nat},\mathcal{D}}$ is the entropy of the predicted velocity distribution at the cached sampling step and $\beta > 0$.

After computing $\mathcal{L}_{\text{A}^2\text{-GRPO},\mathcal{D}}$, EXPLORE updates the trainable parameters with a single optimization step using step size η and gradient-norm clipping threshold $\delta_{\max}$, and the policy is then returned to evaluation mode to continue the rollout. Owing to this in-loop conversion of physical feedback into a step-wise policy improvement signal, the proposed EXPLORE achieves self-improvement entirely at test time, without offline supervised fine-tuning, external critics, or reference-policy caching.

4 Experiments

Benchmarks. Following the official evaluation protocols of UniVLA [69] and ACG [55], we evaluate EXPLORE on three standard VLA benchmarks. **SimplerEnv-WidowX** [39] assesses VLA generalization under variations in lighting, textures, and viewpoints on 4 representative tasks: *Put Spoon on Towel*, *Put Carrot on Plate*, *Stack Block*, and *Put Eggplant in Basket*. DexMimicGen (**DexMG**) [30] is a large-scale automated dataset covering 9 dexterous and bimanual tasks across three robot embodiments to evaluate the cross-embodiment generalization of VLA models. **RoboCasa** [51] is a large-scale simulation benchmark for everyday robotic tasks, comprising 24 atomic tabletop tasks across seven skill families, including pick-and-place, insertion, and drawer manipulation. On all three benchmarks, we report the *average success rate* (**average SR**) over all tasks.

Implementation Details. We instantiate EXPLORE on two representative VLA models, **UniVLA** [69], an autoregressive VLA with FAST action tokenization [56], and **ACG** [55], a diffusion-based VLA with a DiT-style action head and test-time action guidance. We fix the candidate-group size to $K = 8$ across all experiments and keep the action-chunk horizon H aligned with each baseline's default ($H = 5$ for SimplerEnv-WidowX, and $H = 16$ for DexMG and RoboCasa). For reward-margin gating, we set $m_{\text{gate}} = 0.05$ for UniVLA and $m_{\text{gate}} = 0.01$ for ACG. Our A^2-GRPO inherits the optimization scheme of GRPO [58], using default hyperparameters η, and $\delta_{\max}$. For

Table 2: **Results (%) on SimplerEnv-WidowX** across four manipulation tasks. † indicates our re-evaluation using the officially released source.

VLA Models	Spoon on Towel	Carrot on Plate	Stack Blocks	Eggplant in Basket	Overall
Vanilla VLA models					
OpenVLA [33]	0.0	0.0	0.0	4.1	1.0
RT-1-X [87]	0.0	4.2	0.0	0.0	1.1
Octo-Base [53]	12.5	8.3	0.0	43.1	16.0
Octo-Small [53]	47.2	9.7	4.2	56.9	29.5
RoboVLMs [38]	45.8	20.8	4.2	79.2	37.5
SpatialVLA [57]	16.7	25.0	29.2	**100.0**	42.7
Post-Trained VLA models					
UniVLA [69] (Baseline)	83.3	66.7	33.3	95.8	69.8
UniVLA† (Baseline)	79.2	58.3	45.8	95.8	69.8
ContextVLA [25]	60.7	**81.3**	**78.7**	62.0	70.7
InternVLA-M1 [12]	87.5	67.9	31.3	**100.0**	71.7
MemoryVLA [59]	75.0	75.0	37.5	**100.0**	71.9
LoLA [67]	**95.8**	58.3	54.2	79.2	71.9
DepthVLA [78]	75.8	71.7	62.5	89.2	74.8
Xiaomi-Robotics-0 [9]	**95.8**	62.5	75.0	83.3	79.2
Test-Time Adapted VLA models					
BLURR-Pi-0 [50]	91.0	54.0	46.0	93.0	71.0
ExploreVLA (Ours)	87.5	79.2	62.5	95.8	**81.3**

Table 3: **Results (%) on DexMG.** We follow the official DexMG evaluation protocol and report the mean and standard deviation over three random seeds.

VLA Models	GR00T-N1 [4]	ACG [55] (Baseline)	ACG† (Baseline)	**ExploreVLA (Ours)**
Average SR	40.60 ± 3.08	44.00 ± 2.41	36.27 ± 1.15	**46.19** $\pm$**1.86**

evaluation, we follow the official protocols: UniVLA on SimplerEnv-WidowX uses its standard protocol, while results for ACG on RoboCasa and DexMG are averaged over three random seeds. All experiments are run on a single NVIDIA RTX PRO 6000 Blackwell GPU with 96 GB of memory.

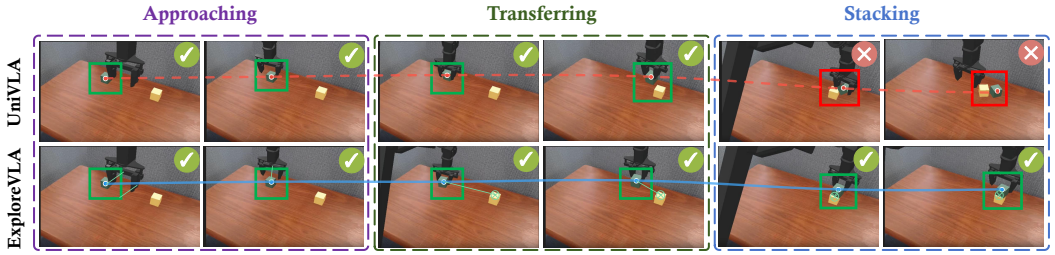
4.1 Main Results

Comparison with Test-Time Adaptation Methods. As shown in Table 1, we compare EXPLORE against three representative test-time adaptation strategies (Best-of-N, LAME-style [6], and Tent-style [65]) on DexMG and SimplerEnv-WidowX. Results show that confidence- and entropy-based methods yield only marginal improvements on DexMG (+1.69 to +6.17) and even degrade performance on SimplerEnv-WidowX, with LAME-style and Tent-style suffering drops of 3.12 and 2.08 points, respectively. In contrast, EXPLORE consistently achieves substantial gains over the baseline, improving the average SR by 9.92 points on DexMG (46.19%) and 11.46 points on SimplerEnv-WidowX (81.25%). These results demonstrate that our physics-grounded process reward, coupled with online RL-based optimization, delivers more effective and reliable test-time adaptation while maintaining a reasonable inference cost. (Further details are provided in Sec. A.2.)

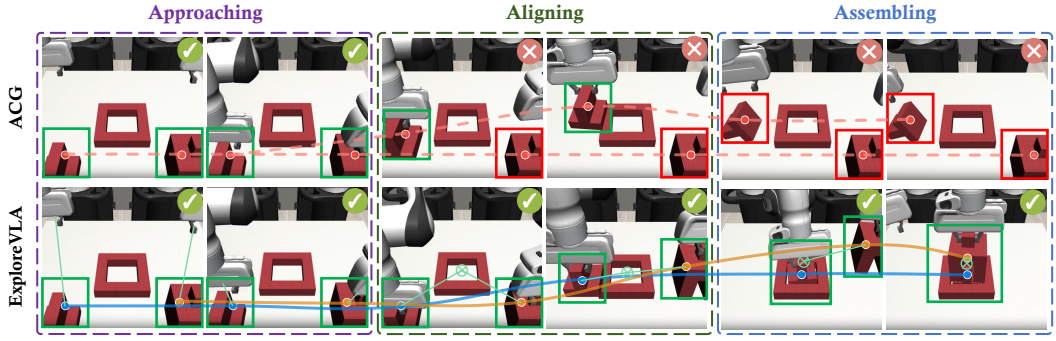
Comparison with State-of-the-Art VLA Models. Results across three benchmarks are summarized in Tables 2 to 4. Overall, ExploreVLA consistently improves strong VLA baselines regardless of action-head architectures across benchmarks. Specifically, ExploreVLA achieves an overall performance of 81.3%, surpassing the strongest post-trained model, Xiaomi-Robotics-0 [9] (79.2%) by 2.1 points and outperforming the test-time adapted BLURR-Pi-0 [50] (71.0%) by a substantial margin of 10.3 points on SimplerEnv-WidowX. On RoboCasa and DexMG, ExploreVLA likewise attains the highest average SR of 43.8% and 46.19%, surpassing the strongest competitors GR00T-N1-DiG [82] (43.2%) and ACG [55] (44.00%), respectively. These

Table 4: **Results (%) on RoboCasa.**

VLA Models	Average SR
GR00T-N1-2B [4]	32.1
BC [28]	32.3
π_0 [5]	38.8
GWM (RoboCasa Human-50) [48]	39.3
GR00T-N1 + DreamGen (100 demos) [26]	39.9
ACG [55] (Baseline)	39.3
ACG† (Baseline)	41.2
π_0 + Pri4R [32]	42.2
GR00T-N1-DiG [82]	43.2
ExploreVLA (Ours)	**43.8**



(a) **Task Instruction:** ***Stack the green block on the yellow block.***



(b) **Task Instruction:** ***Assemble the three pieces to form a block.***

Figure 2: **Qualitative comparison between ExploreVLA and the baselines.** (a) ExploreVLA maintains object-center alignment throughout the stacking task, whereas UniVLA drifts during transfer yet still proceeds with the placement. (b) ExploreVLA's physics feedback enables coordinated bimanual actions that complete the assembly, while ACG fails to synchronize the two arms.

results collectively demonstrate that ExploreVLA achieves new state-of-the-art performance across diverse manipulation scenarios, confirming the effectiveness and generalizability of our EXPLORE for VLA self-improvement at test time.

Qualitative Comparison. Figure 2 presents two representative rollout visualizations. ExploreVLA preserves precise object-center alignment throughout transfer and placement in the stacking task (Figure 2a) and produces well-coordinated bimanual actions that successfully mate the two parts in the assembly task (Figure 2b). As shown by the alignment cues and motion traces, physics-grounded process rewards favor chunks that preserve object-center alignment in stacking and coordinated two-part control in assembly. These reward-guided choices ultimately provide useful feedback for online refinement. (Additional failure cases are provided in Sec. A.3.)

4.2 Ablation Studies

In this section, all ablation studies are conducted under two default settings: (*i*) UniVLA (autoregressive head) on SimplerEnv-WidowX and (*ii*) ACG (diffusion head) on DexMG.[2] Additional ablations on *reward-margin gating* and hyperparameter settings are provided in Sec. A.1.

Effect of Coordinated Exploration-Exploitation Action-Chunk Generation. Table 5 compares EXPLORE with three alternative action-chunk generation strategies under each default setting. For setting (*i*), we consider: ▷ *Structured Prefix-Diverse*, which generates more fixed-length action-token sequences with different sampling profiles while applying diversity penalties to early action-prefix tokens; ▷ *Temperature-Stratified*, yielding various candidates by sampling at a distinct temperature; and ▷ *Stochastic Latent*, drawing more candidates via high-confidence sampling in the VLA's prior latent space. For setting (*ii*), we compare: ▷ *Guidance-Scale*, generating diverse candidate action chunks by varying the ACG guidance scale; ▷ *Inference-Step*, obtaining different candidates by varying the number of flow-matching inference steps; and ▷ *Noise-Driven*, generating various candidates by injecting different initial noise seeds or local noise perturbations into the flow-matching sampler. From the results in Table 5, we draw two key observations. ▶ Compared to the baseline, all action-space expansion strategies yield consistent improvements (+4.62 to +11.46 points), confirming that

[2]For fair comparison, we hereafter report performance gains over re-evaluated baselines under the same evaluation pipeline.

Table 5: **Comparison between action-chunk generation strategies for action-space expansion.**

Methods	Key Idea for Candidate Generation	Average SR (%)
UniVLA (Baseline)	/	69.79
Structured Prefix-Diverse Strategy	Fixed-length action blocks with profile annealing	77.08 (+ 7.29)
Temperature-Stratified Strategy	Generate candidates with different temperatures	78.13 (+ 8.34)
Stochastic Latent Strategy	Isotropic Gaussian perturbation from the latent prior	80.21 (+ 10.42)
Coordinated E-E Generation	**Combine exploitation and exploration groups**	**81.25** (+ 11.46)
ACG (Baseline)	/	36.27
Guidance-Scale Strategy	Adjust ACG guidance scale	41.20 (+ 4.93)
Inference-Step Strategy	Change the number of inference steps	40.89 (+ 4.62)
Noise-Driven Strategy	Independent Gaussian noise initialization	45.22 (+ 8.95)
Coordinated E-E Generation	**Combine exploitation and exploration groups**	**46.19** (+ 9.92)

broadening the coverage of effective actions is fundamental to unleashing the potential of test-time RL for VLA models. ▶ Compared with these alternatives, EXPLORE achieves the best performance under both settings, reaching 81.25% and 46.19%. These gains demonstrate that explicitly coordinating exploitation and exploration groups produces more diverse yet purposeful action candidates than purely stochastic perturbations or single-axis variations, validating the effectiveness of EXPLORE.

Impact of $\mathcal{G}_{\mathcal{D}}^{\text{exploit}}$ and $\mathcal{G}_{\mathcal{D}}^{\text{explore}}$. Table 6 further analyzes the contributions of the two action-chunk groups defined in Eq. (3). The results show that exploitation-only consistently outperforms exploration-only under both default settings (80.21% vs. 73.96%; 45.22% vs. 44.44%), indicating that high-confidence candidates provide a more reliable foundation than purely exploratory ones. Importantly, combining both groups achieves the best performance, confirming that $\mathcal{G}_{\mathcal{D}}^{\text{exploit}}$ and $\mathcal{G}_{\mathcal{D}}^{\text{explore}}$ are complementary. Exploitation anchors stability, while exploration broadens coverage of effective actions.

Table 6: **Ablation on candidate action-chunk group.**

Methods	$\mathcal{G}_{\mathcal{D}}^{\text{exploit}}$	$\mathcal{G}_{\mathcal{D}}^{\text{explore}}$	UniVLA	ACG
Exploitation-Only Generation	✓	✗	80.21	45.22
Exploration-Only Generation	✗	✓	73.96	44.44
Coordinated E-E Generation	✓	✓	**81.25**	**46.19**

Effect of Physics-Grounded Process Reward. Table 7 compares our reward design against three alternatives: ▷ *Semantic Alignment*, which scores each candidate action chunk by the cosine similarity between its resulting observation and the given task instruction; ▷ *Temporal Consistency*, scoring action chunks with smooth short-horizon dynamics by penalizing abrupt changes between consecutive actions; and ▷ *Intra-Group Consensus*, where each candidate is scored using its agreement with the remaining action chunks in the same group. From the results, we observe that while all three alternatives provide only marginal gains over the baseline (+1.04 to +3.39 points), our process reward delivers substantially larger improvements of +11.46 and +9.92 points under both default settings, surpassing the strongest alternative by over 6 points. This pronounced gap suggests that dense physical interaction feedback directly reflects the actual outcomes of executed actions, offering more informative supervision for effective test-time adaptation.

Table 7: **Comparison of reward designs.** Physics-grounded process rewards preserve informative feedback and yield the best test-time adaptation performance.

Methods	Key Signal in Reward	UniVLA	ACG
Baseline	/	69.79	36.27
Semantic Alignment Reward	Text-image alignment	71.88 (+ 2.09)	37.81 (+ 1.54)
Temporal Consistency Reward	Intra-sequence smoothness	72.08 (+ 2.29)	39.66 (+ 3.39)
Intra-Group Consensus Reward	Group agreement across samples	70.83 (+ 1.04)	39.66 (+ 3.39)
Physics-Grounded Process Reward	Dense physical interaction feedback	**81.25** (+ 11.46)	**46.19** (+ 9.92)

5 Conclusion

In this work, we investigate test-time reinforcement learning for VLA models and propose EXPLORE, a framework for simulator-based online VLA self-improvement. Within EXPLORE, exploitation-exploration action-chunk generation expands the effective action space, while physics-grounded process rewards provide feedback for online policy adaptation through A^2-GRPO. Extensive evaluations on SimplerEnv-WidowX, RoboCasa, and DexMG demonstrate the effectiveness of EXPLORE in improving both autoregressive and diffusion-based VLA backbones across diverse tasks.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant Nos. 62322113, 62376156), as well as the Shanghai Municipal Special Program for Basic Research on General AI Foundation Models (Grant No. 2025SHZDZX025G15).

References

[1] Ekin Akyurek, Mehul Damani, Linlu Qiu, Han Guo, Yoon Kim, and Jacob Andreas. The surprising effectiveness of test-time training for abstract reasoning. *arXiv:2411.07279*, 2024. 3

[2] Zechen Bai, Chen Gao, and Mike Zheng Shou. Evolve-vla: Test-time training from environment feedback for vision-language-action models. *arXiv:2512.14666*, 2025. 1, 2, 3

[3] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv:2501.00663*, 2024. 3

[4] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv:2503.14734*, 2025. 2, 7

[5] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. In *RSS*, 2025. 2, 7

[6] Malik Boudiaf, Romain Mueller, Ismail Ben Ayed, and Luca Bertinetto. Parameter-free online test-time adaptation. In *CVPR*, pages 8344–8353, 2022. 3, 6, 7, 23

[7] 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:2212.06817*, 2022. 1, 2

[8] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv:2407.21787*, 2024. 1

[9] Rui Cai, Jun Guo, Xinze He, Piaopiao Jin, Jie Li, Bingxuan Lin, Futeng Liu, Wei Liu, Fei Ma, Kun Ma, et al. Xiaomi-robotics-0: An open-sourced vision-language-action model with real-time execution. *arXiv:2602.12684*, 2026. 2, 7

[10] Chi-Lam Cheang, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, et al. Gr-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv:2410.06158*, 2024. 1, 2

[11] Mark Chen, Jerry Tworek, Heewoo Jun, et al. Evaluating large language models trained on code. *arXiv:2107.03374*, 2021. 1

[12] Xinyi Chen, Yilun Chen, Yanwei Fu, Ning Gao, Jiaya Jia, Weiyang Jin, Hao Li, Yao Mu, Jiangmiao Pang, Yu Qiao, et al. Internvla-m1: A spatially guided vision-language-action framework for generalist robot policy. *arXiv:2510.13778*, 2025. 2, 7

[13] Yuxuan Chen and Xiao Li. Rlrc: Reinforcement learning-based recovery for compressed vision-language-action models. *arXiv:2506.17639*, 2025. 2

[14] 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. *arXiv:2506.08440*, 2025. 2

[15] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, et al. Training verifiers to solve math word problems. *arXiv:2110.14168*, 2021. 1

[16] Karan Dalal, Daniel Koceja, Jiarui Xu, Yue Zhao, Shihao Han, Ka Chun Cheung, Jan Kautz, Yejin Choi, Yu Sun, and Xiaolong Wang. One-minute video generation with test-time training. In *CVPR*, pages 17702–17711, 2025. 3

[17] Haikang Deng and Colin Raffel. Reward-augmented decoding: Efficient controlled text generation with a unidirectional reward model. In *EMNLP*, pages 11781–11791, 2023. 3

[18] Pengxiang Ding, Jianfei Ma, Xinyang Tong, Binghong Zou, Xinxin Luo, Yiguo Fan, Ting Wang, Hongchao Lu, Panzhong Mo, Jinxin Liu, et al. Humanoid-vla: Towards universal humanoid control with visual integration. *arXiv:2502.14795*, 2025. 2

[19] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv:2303.03378*, 2023. 1, 2

[20] Moritz Hardt and Yu Sun. Test-time training on nearest neighbors for large language models. *arXiv:2305.18466*, 2023. 3

[21] Xuefeng Hu, Ke Zhang, Min Sun, Albert Chen, Cheng-Hao Kuo, and Ram Nevatia. Bafta: Backprop-free test-time adaptation for zero-shot vision-language models. *arXiv:2406.11309*, 2024. 3

[22] Dongchi Huang, Zhirui Fang, Tianle Zhang, Yihang Li, Lin Zhao, and Chunhe Xia. Co-rft: Efficient fine-tuning of vision-language-action models through chunked offline reinforcement learning. *arXiv:2508.02219*, 2025. 2

[23] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv:2504.16054*, 2025. 2

[24] Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. In *NeurIPS*, pages 2427–2440, 2021. 3

[25] Huiwon Jang, Sihyun Yu, Heeseung Kwon, Hojin Jeon, Younggyo Seo, and Jinwoo Shin. Contextvla: Vision-language-action model with amortized multi-frame context. *arXiv:2510.04246*, 2025. 2, 7

[26] Joel Jang, Seonghyeon Ye, Zongyu Lin, Jiannan Xiang, Johan Bjorck, Yu Fang, Fengyuan Hu, Spencer Huang, Kaushil Kundalia, Yen-Chen Lin, et al. Dreamgen: Unlocking generalization in robot learning through video world models. *arXiv:2505.12705*, 2025. 2, 7

[27] Suhyeok Jang, Dongyoung Kim, Changyeon Kim, Youngsuk Kim, and Jinwoo Shin. Verifier-free test-time sampling for vision language action models. *arXiv:2510.05681*, 2025. 2

[28] Xiaogang Jia, Qian Wang, Anrui Wang, Han A. Wang, Balázs Gyenes, Emiliyan Gospodinov, Xinkai Jiang, Ge Li, Hongyi Zhou, Weiran Liao, et al. Pointmappolicy: Structured point cloud processing for multi-modal imitation learning. *arXiv:2510.20406*, 2025. 7

[29] Jingjing Jiang, Chao Ma, Xurui Song, Hanwang Zhang, and Jun Luo. Corvid: Improving multimodal large language models towards chain-of-thought reasoning. In *CVPR*, pages 3034–3046, 2025. 1

[30] Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Jim Fan, and Yuke Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning. In *ICRA*, pages 16923–16930, 2025. 2, 6

[31] Adilbek Karmanov, Dayan Guan, Shijian Lu, Abdulmotaleb El Saddik, and Eric Xing. Efficient test-time adaptation of vision-language models. In *CVPR*, pages 14162–14171, 2024. 3

[32] Jisoo Kim, Jungbin Cho, Sanghyeok Chu, Ananya Bal, Jinhyung Kim, Gunhee Lee, Sihaeng Lee, Seung Hwan Kim, Bohyung Han, Hyunmin Lee, et al. Pri4r: Learning world dynamics for vision-language-action models with privileged 4d representation. *arXiv:2603.01549*, 2026. 2, 7

[33] 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. *arXiv:2406.09246*, 2024. 2, 7

[34] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *RSS*, 2025. 2

[35] Jacky Kwok, Christopher Agia, Rohan Sinha, Matt Foutter, Shulu Li, Ion Stoica, Azalia Mirhoseini, and Marco Pavone. Robomonkey: Scaling test-time sampling and verification for vision-language-action models. *arXiv:2506.17811*, 2025. 1, 2

[36] Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, 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. *arXiv:2509.09674*, 2025. 2

[37] Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv:2411.19650*, 2024. 2

[38] Xinghang Li, Peiyan Li, Minghuan Liu, Dong Wang, Jirong Liu, Bingyi Kang, Xiao Ma, Tao Kong, Hanbo Zhang, and Huaping Liu. Towards generalist robot policies: What matters in building vision-language-action models. *arXiv:2412.14058*, 2024. 2, 7

[39] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, et al. Evaluating real-world robot manipulation policies in simulation. *arXiv:2405.05941*, 2024. 2, 6

[40] Yang Li, Naiyan Wang, Jianping Shi, Jingying Liu, Xiaodi Hou, and Alex C. Kot. Revisiting batch normalization for practical domain adaptation. *arXiv:1603.04779*, 2016. 3

[41] Yujia Li, David Choi, Junyoung Chung, et al. Competition-level code generation with AlphaCode. *Science*, 378(6624):1092–1097, 2022. 1

[42] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *ICLR*, 2024. 1

[43] Fanqi Lin, Ruiqian Nai, Yingdong Hu, Jiacheng You, Junming Zhao, and Yang Gao. Onetwovla: A unified vision-language-action model with adaptive reasoning. *arXiv:2505.11917*, 2025. 2

[44] Changyu Liu, Yiyang Liu, Taowen Wang, Qiao Zhuang, James Chenhao Liang, Wenhao Yang, Renjing Xu, Qifan Wang, Dongfang Liu, and Cheng Han. On-the-fly vla adaptation via test-time reinforcement learning. *arXiv:2601.06748*, 2026. 1, 2, 3

[45] Jiaming Liu, Hao Chen, Pengju An, Zhuoyang Liu, Renrui Zhang, Chenyang Gu, Xiaoqi Li, Ziyu Guo, Sixiang Chen, Mengzhen Liu, et al. Hybridvla: Collaborative diffusion and autoregression in a unified vision-language-action model. *arXiv:2503.10631*, 2025. 2

[46] Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv:2410.07864*, 2024. 2

[47] Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. Vla-rl: Towards masterful and general robotic manipulation with scalable reinforcement learning. *arXiv:2505.18719*, 2025. 2

[48] Guanxing Lu, Baoxiong Jia, Puhao Li, Yixin Chen, Ziwei Wang, Yansong Tang, and Siyuan Huang. Gwm: Towards scalable gaussian world models for robotic manipulation. In *ICCV*, 2025. 7

[49] Xiaosong Ma, Jie Zhang, Song Guo, and Wenchao Xu. Swapprompt: Test-time prompt adaptation for vision-language models. In *NeurIPS*, pages 65252–65264, 2023. 3

[50] Xiaoyu Ma, Zhengqing Yuan, Zheyuan Zhang, Kaiwen Shi, Lichao Sun, and Yanfang Ye. Blurr: A boosted low-resource inference for vision-language-action models. *arXiv:2512.11769*, 2025. 2, 7

[51] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *RSS*, 2024. 2, 6

[52] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *ICML*, pages 16888–16905, 2022. 3

[53] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *RSS*, 2024. 2, 7

[54] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *ICRA*, pages 6892–6903, 2024. 1, 2

[55] Minho Park, Kinam Kim, Junha Hyung, Hyojin Jang, Hoiyeong Jin, Jooyeol Yun, Hojoon Lee, and Jaegul Choo. Acg: Action coherence guidance for flow-based vla models. In *ICRA*, 2026. 2, 6, 7

[56] 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:2501.09747*, 2025. 6

[57] Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. In *RSS*, 2025. 1, 2, 7

[58] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv:2402.03300*, 2024. 5, 6

[59] Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. *arXiv:2508.19236*, 2025. 2, 7

[60] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv:2506.01844*, 2025. 2

[61] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv:2408.03314*, 2024. 1, 3

[62] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, pages 9229–9248, 2020. 3

[63] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv:2407.04620*, 2024. 3

[64] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. MIT Press, second edition, 2018. 1

[65] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno A. Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *ICLR*, 2021. 3, 6, 7, 23

[66] Renhao Wang, Yu Sun, Arnuv Tandon, Yossi Gandelsman, Xinlei Chen, Alexei A Efros, and Xiaolong Wang. Test-time training on video streams. *JMLR*, (9):1–29, 2025. 3

[67] Xiaofan Wang, Xingyu Gao, Jianlong Fu, Zuolei Li, Dean Fortier, Galen Mullins, Andrey Kolobov, and Baining Guo. Lola: Long horizon latent action learning for general robot manipulation. *arXiv:2512.20166*, 2025. 2, 7

[68] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *ICLR*, 2023. 3

[69] Yuqi Wang, Xinghang Li, Wenxuan Wang, Junbo Zhang, Yingyan Li, Yuntao Chen, Xinlong Wang, and Zhaoxiang Zhang. Unified vision-language-action model. *arXiv:2506.19850*, 2025. 2, 6, 7

[70] Yuqing Wen, Hebei Li, Kefan Gu, Yucheng Zhao, Tiancai Wang, and Xiaoyan Sun. Llada-vla: Vision language diffusion action models. *arXiv:2509.06932*, 2025. 2

[71] Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. *arXiv:2312.13139*, 2023. 1, 2

[72] Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv:2408.00724*, 2024. 1

[73] Yilin Wu, Anqi Li, Tucker Hermans, Fabio Ramos, Andrea Bajcsy, and Claudia Pérez-D'Arpino. Do what you say: Steering vision-language-action models via runtime reasoning-action alignment verification. *arXiv:2510.16281*, 2025. 2

[74] Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P. Lillicrap, Kenji Kawaguchi, and Michael Shieh. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv:2405.00451*, 2024. 3

[75] Jianwei Yang, Reuben Tan, Qianhui Wu, Ruijie Zheng, Baolin Peng, Yongyuan Liang, Yu Gu, Mu Cai, Seonghyeon Ye, Joel Jang, Yuquan Deng, Lars Liden, and Jianfeng Gao. Magma: A foundation model for multimodal ai agents. In *CVPR*, pages 20320–20331, 2025. 2

[76] Siyuan Yang, Yang Zhang, Haoran He, Ling Pan, Xiu Li, Chenjia Bai, and Xuelong Li. Steering vision-language-action models as anti-exploration: A test-time scaling approach. *arXiv:2512.02834*, 2025. 1

[77] Angen Ye, Zeyu Zhang, Boyuan Wang, Xiaofeng Wang, Dapeng Zhang, and Zheng Zhu. Vla-r1: Enhancing reasoning in vision-language-action models. *arXiv:2510.01623*, 2025. 2

[78] Tianyuan Yuan, Yicheng Liu, Chenhao Lu, Zhuoguang Chen, Tao Jiang, and Hang Zhao. Depthvla: Enhancing vision-language-action models with depth-aware spatial reasoning. *arXiv:2510.13375*, 2025. 2, 7

[79] Zehua Zang, Xi Wang, Fuchun Sun, Xiao Xu, Lixiang Lium, Jiahuan Zhou, and Jiangmeng Li. Test-time perturbation learning with delayed feedback for vision-language-action models. In *CVPR*, 2026. 1, 2, 3

[80] Hongyin Zhang, Zifeng Zhuang, Han Zhao, Pengxiang Ding, Hongchao Lu, and Donglin Wang. Reinbot: Amplifying robot visual-language manipulation with reinforcement learning. *arXiv:2505.07395*, 2025. 2

[81] Kaiyan Zhang, Jiayuan Zhang, Haoxin Li, Xuekai Zhu, Ermo Hua, Xingtai Lv, Ning Ding, Biqing Qi, and Bowen Zhou. Openprm: Building open-domain process-based reward models with preference trees. In *ICLR*, 2025. 3

[82] Wanpeng Zhang, Ye Wang, Hao Luo, Haoqi Yuan, Yicheng Feng, Sipeng Zheng, Qin Jin, and Zongqing Lu. Dig-flow: Discrepancy-guided flow matching for robust vla models. *arXiv:2512.01715*, 2025. 2, 7

[83] Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 3d-vla: 3d vision-language-action generative world model. *arXiv:2403.09631*, 2024. 2

[84] Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. *arXiv:2412.10345*, 2024. 2

[85] Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. *arXiv:2310.04406*, 2023. 3

[86] Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv:2504.10479*, 2025. 1

[87] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *CoRL*, pages 2165–2183, 2023. 1, 2, 7

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect our paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations and representative failure cases in Appendix A.3.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: This paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide all implementation details.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will soon release our code.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide all the experimental details.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We use official evaluation protocols provided by the corresponding benchmarks.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide all the implementation details.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We have checked our work.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss this in the conclusion and introduction sections.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All are properly referred to.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No such experiments.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No such experiments.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLM is used for paper polishing.

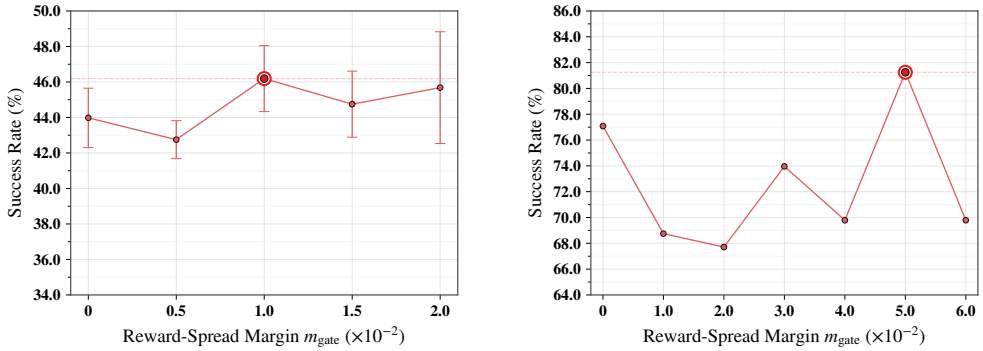
Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.

A Appendix

A.1 Additional Ablation Study

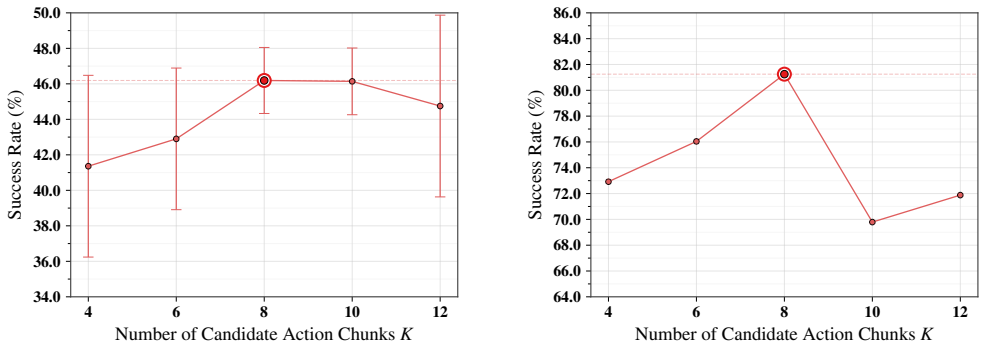
Impact of Reward-Margin Gating. We ablate the reward-margin gate threshold m_{gate} in Figure 3. A moderate threshold achieves the best performance on both DexMG and SimplerEnv-WidowX. When m_{gate} is too small, A^2-GRPO may admit updates from weak reward margins among candidate action chunks, introducing noisy gradients into test-time policy optimization. When m_{gate} is too large, many reward-discriminative candidate groups are skipped, limiting the conversion of physics-grounded process rewards into online policy improvement. These results show that reward-margin gating is important for filtering unreliable group feedback and stabilizing A^2-GRPO updates.



(a) DexMG. (b) SimplerEnv-WidowX.

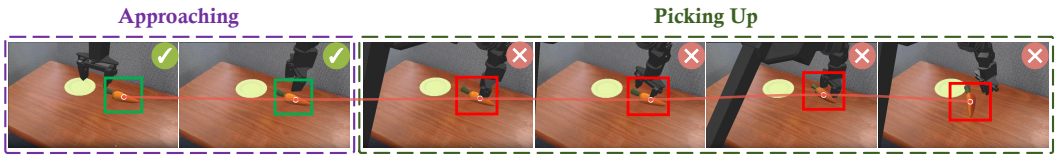
Figure 3: **Effect of the reward-margin gating threshold m_{gate}.** We vary m_{gate} on DexMG and SimplerEnv-WidowX. Highlighted points indicate the default settings used in the main experiments, and error bars on DexMG denote standard deviations over three random seeds.

Impact of the Number of Candidate Action Chunks. We further vary the number of candidate action chunks K in Figure 4. Increasing K improves performance at first, since a larger candidate group broadens the effective action-space coverage and provides more reward-discriminative chunks for physics-grounded evaluation. However, performance saturates or slightly declines when K becomes too large, as additional candidates introduce more redundant rollouts and increase the cost of process-reward evaluation. Overall, $K = 8$ provides a strong balance between test-time adaptation performance and computational efficiency.

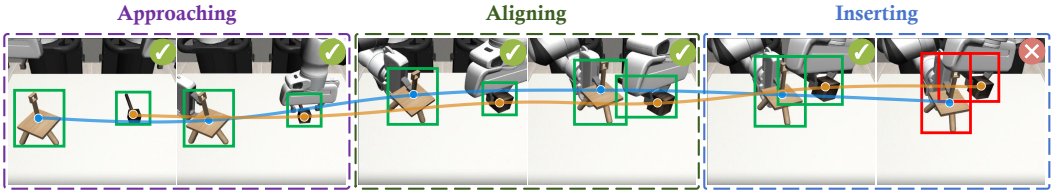


(a) DexMG. (b) SimplerEnv-WidowX.

Figure 4: **Effect of the candidate-group size K.** We vary K on DexMG and SimplerEnv-WidowX. Highlighted points indicate the default settings used in the main experiments, and error bars on DexMG denote standard deviations over three random seeds.



(a) **Task Instruction:** ***Put the carrot on the plate.***



(b) **Task Instruction:** ***Pick up the thread and insert it into the ring.***

Figure 5: **Failure case study of ExploreVLA.** (a) ExploreVLA approaches and contacts the carrot but fails to establish a secure grasp during pick-up, leaving the carrot on the table instead of transferring it to the plate. (b) ExploreVLA aligns the thread with the ring direction, but the ring slightly shifts while being held by the other arm, resulting in an unsuccessful insertion.

A.2 Additional Comparisons with Test-Time Adaptation Methods

Details on Compared Test-Time Adaptation Methods. We compare EXPLORE with representative test-time strategies in two categories. Local self-supervised adaptation, represented by Tent-style, updates the VLA from the current test input to mitigate distribution shift, but lacks external feedback on action quality. Inference-time selection, represented by Best-of-N and LAME-style, selects an action chunk at the current decision step, but does not accumulate experience through parameter updates. For a fair comparison, we perform hyperparameter search on their key hyperparameters (KHP) and report their best performance. All methods share the same rollout interface and backbone-specific action decoding pipeline as EXPLORE. Specifically,

- **Best-of-N** removes A^2-GRPO and directly executes the highest-scoring action chunk among sampled candidates. KHP is the number of candidate action chunks (N).
- **LAME-style** [6] smooths candidate scores on a trajectory-distance graph without updating model parameters. For autoregressive heads, the unary score is the average conditional log-probability; for diffusion heads, it is computed from executable action chunks. KHP are the number of candidate action chunks (N), the graph temperature (τ), the number of nearest neighbors (k), and the number of smoothing iterations (T).
- **Tent-style** [65] updates the trainable action head using a single sampled action chunk from the current test input. For autoregressive heads, we optimize constrained action-token likelihood; for diffusion heads, we optimize the cached velocity-matching loss. KHP are the learning rate (η), the gradient-clipping threshold $(\delta_{\max})$, and the loss clipping threshold (ϵ).

More Discussion on Table 1. Table 1 further reveals the efficiency–performance trade-off among different test-time strategies. We analyze the results from two aspects. ▶ In terms of runtime, the cost mainly depends on how each method uses test-time samples. Tent-style is the most efficient baseline, since it updates the action head with only a single sampled action chunk from the current test input (4.49h on DexMG and 1.74h on SimplerEnv-WidowX). Best-of-N incurs higher cost because it samples multiple candidate chunks and directly selects the highest-scoring one without policy updates (14.04h on DexMG and 4.43h on SimplerEnv-WidowX). LAME-style further performs graph-based score smoothing over sampled candidates, leading to the largest baseline runtime on DexMG and SimplerEnv-WidowX (32.74h and 6.93h). EXPLORE requires additional computation for physics-grounded candidate evaluation and online A^2-GRPO updates, resulting in 16.84h on DexMG and 7.61h on SimplerEnv-WidowX. ▶ In terms of average SR, the key factor is whether the test-time signal is physics-grounded and used for policy updates. Tent-style updates the action head online but relies on prediction confidence rather than physics-grounded action quality, leading to only +1.69 points on DexMG and a 2.08-point drop on SimplerEnv-WidowX. Best-of-N uses physics-grounded scores for candidate selection, but without policy updates, and it brings moderate improvements (+6.17 and +3.13 points). LAME-style smooths candidate scores for selection, but also

lacks policy updates, yielding +1.69 points on DexMG while degrading SimplerEnv-WidowX by 3.12 points. EXPLORE combines physics-grounded candidate evaluation with online A^2-GRPO updates, achieving the best average SR (+9.92 and +11.46 points). Overall, EXPLORE has a reasonable runtime compared with selection-based methods, while enabling actual policy refinement.

A.3 Limitations and Failure Cases

Failure Case. We present two representative failure cases of ExploreVLA in Figure 5. Although physics-grounded process rewards provide dense feedback on spatial alignment, contact relations, and manipulation progress, they cannot fully resolve failures caused by challenging object geometry or unstable interaction dynamics. In the carrot placement case, Figure 5a shows that ExploreVLA reaches the carrot and makes contact with it, but the grasp remains unstable. The elongated and irregular object geometry makes the outcome highly sensitive to the exact contact location. As a result, ExploreVLA fails to establish a stable grasp before lifting, and the carrot remains on the table. The thread insertion failure in Figure 5b arises from a different challenge. Although ExploreVLA brings the thread close to the ring and roughly aligns it with the insertion direction, the arm holding the ring is not completely stationary. Small residual motion changes the ring pose during insertion, causing the final insertion to fail despite the initial alignment. These observations highlight remaining limitations in stable grasp establishment and bimanual object holding during contact-rich manipulation.

Limitations. Although EXPLORE achieves great improvements across different VLA backbones and benchmarks, it still has several limitations in challenging manipulation scenarios. As illustrated in Sec. A.3, the physics-grounded process reward is built from task-relevant physical primitives, but it does not explicitly model fine-grained object shape, making elongated or irregular objects sensitive to contact locations and grasp stability. Moreover, EXPLORE derives supervision from interaction outcomes, while the intrinsic stability of the action chunk itself is only indirectly captured, so residual motions may still prevent the manipulated object from settling for subsequent precise actions. In addition, as discussed in Sec. A.2, the performance gains of EXPLORE come with additional test-time computation, suggesting that more efficient optimization algorithms remain worth exploring.