On-the-Fly VLA Adaptation via Test-Time Reinforcement Learning

Anonymous ACL submission

Abstract

Vision-Language-Action (VLA) models have recently emerged as a powerful paradigm for general-purpose robot learning, enabling agents to map visual observations and natural-language instructions into executable robotic actions. Though popular, they are primarily trained via supervised fine-tuning or training-time reinforcement learning, requiring explicit fine-tuning phases, human interventions, or controlled data collection. Consequently, existing methods remain unsuitable for challenging simulated- or physical-world deployments, where robots must respond autonomously and flexibly to evolving environments. To address this limitation, we introduce a Test-Time Reinforcement Learning for VLAs (TT-VLA), a framework that enables on-the-fly policy adaptation during inference. TT-VLA formulates a dense reward mechanism that leverages step-by-step task-progress signals to refine action policies during test time while preserving the SFT/RL-trained priors, making it an effective supplement to current VLA models. Empirical results show that our approach enhances overall adaptability, stability, and task success in dynamic, previously unseen scenarios under simulated and real-world settings. We believe TT-VLA offers a principled step toward self-improving, deployment-ready VLAs.

1 Introduction

The ability to adapt to changing conditions is a fundamental requirement for intelligent agents operating in the real world. However, agents trained in fixed, well-defined environments frequently fail when confronted with the inherent real-world dynamic variability (Dulac-Arnold et al., 2021; Tambe et al., 1995; Kormushev et al., 2013). This gap between static training regimes and dynamic deployment environments remains a central challenge in robotics and embodied intelligence.

Current Vision-Language-Action (VLA) models stand at the heart of this challenge. These VLAs integrate perception, language understanding, and control to translate visual observations and natural language instructions into executable actions, representing a significant step and performance gains toward general-purpose embodied intelligence (Kim et al., 2024; Brohan et al., 2023; Zitkovich et al., 2023; Wang et al., 2025b; Kwok et al., 2025). Yet, most VLAs remain bound to fixed policies, which are generally trained either through supervised fine-tuning (SFT) or through training-time reinforcement learning (RL) on curated datasets, explicit fine-tuning phases, and controlled environments.

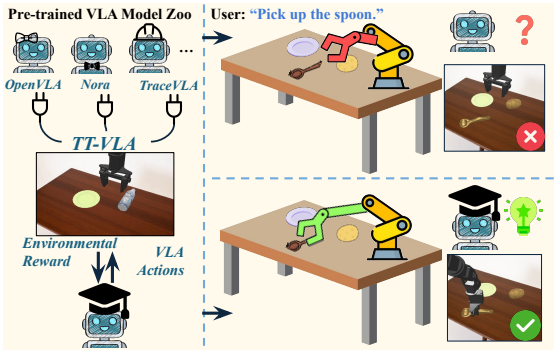


Figure 1: **TT-VLA supplements to SFT/RL-trained VLAs** by continuously adapting policies to environment-derived rewards at test time, improving robustness to distributional shifts without retraining.

In practice, these rigid policies limit VLAs' capacity for challenging simulated-/physical-world applications, where ***robots must adapt at test time as conditions evolve or distribution shifts***. In a broader perspective, current research in language or vision understanding has begun to explore test-time training (TTT) (Zuo et al., 2025; Li et al., 2025c; Karmanov et al., 2024; Hu et al., 2024; Ma et al., 2023) to update models directly on incoming test streams, often without ground-truth labels, underscoring a promising direction toward flexible model adjustments. These advances have emerged in other domains, leaving VLA test-time adaptation largely underexplored. We further find that current TTT cannot be directly applied to VLAs, as the

multimodal nature brings substantial distributional shifts and is evolving (see §4.5).

In light of this view, we propose a Test-Time Reinforcement Learning framework for VLA (TT-VLA) that performs online inference-time policy fine-tuning efficiently without retraining cycles, preserving SFT/training-time RL priors while closing the loop with dense inference-time reward signals. Here, TTT provides accessibility for test-time adaptation, while RL complements it by addressing substantial variations in environmental conditions and underlying distributions.

Different from traditional session-based RL (*i.e.*, SimpleVLA-RL), we creatively derive dense shaping signals from task-agnostic proxies to timely and effectively utilize the limited test-time information and optimize the VLA policy. These shaping signals operate independently across time frames, enabling stable, versatile, and continuous adjustments. Our design also bridges the prevailing offline-RL/VLA pipeline and the demands of real-world robotics, enabling continuous self-improvement under dynamic, previously unseen conditions. Extensive experiments conducted in both simulated and real-world robotic environments demonstrate that our method can naturally boost the performance of existing SFT-/RL-based approaches.

2 Related Work

2.1 VLA Generalization & Adaptation

Current VLAs (Brohan et al., 2023; Mees et al., 2022; Pong et al., 2020; Kwok et al., 2025) are predominantly optimized via supervised fine-tuning on large, curated triplets (Zitkovich et al., 2023; Kim et al., 2024; Plaat et al., 2024; Jiang et al., 2025b; Kim et al., 2025; Liu et al., 2024a), which yields strong performance in static, well-structured settings but leads to brittle behavior and limited robustness under distributional shifts due to the lack of adaptive learning mechanisms (Shenfeld et al., 2025; Chen et al., 2025a; Xu et al., 2024; Chen et al., 2025b; Wang et al., 2025a).

To address these limitations, recent studies have explored integrating VLAs with reinforcement learning. RL allows policies to actively interact with pre-collected environments or demonstrations, thereby enabling continual adaptation and performance improvement beyond static supervision (Huang et al., 2025; Zhang et al., 2025; Mark et al., 2024; Chen et al., 2025c; Ye et al., 2025; Li et al., 2025a; Lu et al., 2025; Jiang et al., 2025a; Chen and Li, 2025; Wu et al., 2021; Guo et al., 2025b). By incorporating interaction-driven feedback, RL-augmented VLAs refine their behaviors to task-specific objectives (Huang et al., 2025; Zhang et al., 2025; Mark et al., 2024) and environmental variations (Chen et al., 2025c; Ye et al., 2025; Li et al., 2025a; Lu et al., 2025; Jiang et al., 2025a; Chen and Li, 2025; Wu et al., 2021; Li et al., 2025b; Guo et al., 2025b), demonstrating improved sample efficiency and generalization across unseen scenarios. Despite these advances, existing RL approaches still adhere to a training-deployment separation paradigm, without mechanism for inference-time adaptation once the model is deployed. This gap leaves VLAs vulnerable to unanticipated dynamics during testing, where retraining is impractical due to time or data constraints.

2.2 Test-Time Training

Test-time training (TTT) is originally proposed for adapting pre-trained models to a target domain using only unlabeled test data (Sun et al., 2020; Hu et al., 2025; Yoon et al., 2024; Xiao et al., 2025; Zhu et al., 2024; Zuo et al., 2025). Unlike SFT (Jia et al., 2022; Mosbach et al., 2021; Han et al., 2023; Wang et al., 2025a; Liu et al., 2025b, 2024b; Hu et al., 2022; Zeng et al., 2024) or traditional RL (Sutton et al., 1999a; Guo et al., 2025a; Huang et al., 2025; Zhang et al., 2025; Mark et al., 2024), TTT leverages self-supervised objectives (*e.g.*, entropy minimization) to calibrate models during inference, thereby enabling effective domain adaptation in the absence of both human-curated labels and external feedback.

Intuitively, TTT can be naturally extended to VLAs to enable efficient adaptation during inference. However, unlike relatively intuitive single-domain tasks (*e.g.*, vision, language), where pre-trained models exhibit high generalization capability (Brown et al., 2020; Wei et al., 2022; Goyal et al., 2023) and inter-task discrepancies are relatively minor (Hu et al., 2025; Liu et al., 2021; Zhao et al., 2023; Han et al., 2025), robotic tasks often entail substantial distributional shifts and evolving conditions across both visual and linguistic modalities (Pong et al., 2020; Wang et al., 2025a; Kim et al., 2024; Liu et al., 2024a). Consequently, fixed, protocol-driven self-supervised objectives become inadequate (see §4.5) and overly generic.

Recent work has explored reinforcement-based alternatives to the purely self-supervised test-time training objectives. In particular, EVOLVE-

VLA (Bai et al., 2025b) leverages task progress as a reward signal to optimize VLA policies during deployment. However, its reliance on GRPO-style optimization incurs substantial computational overhead, limiting its applicability to real-time robotic settings with strict latency constraints. We provide a more detailed discussion regarding the use of GRPO for TTT in VLAs in Appendix §S7.

To address these limitations, we propose an RL-driven test-time learning framework that enables efficient online adaptation (see §3).

3 Method

In this section, we introduce Test-Time Reinforcement Learning framework for VLA (TT-VLA), a novel test-time training approach designed to enhance VLA performance through on-the-fly policy adaptation. In §3.1, we first provide background on Proximal Policy Optimization (PPO) and VLAs. We then present TT-VLA in §3.2. §3.3 further provides TT-VLA's theoretical analysis and insights. The overall framework is shown in Fig. 2.

3.1 Preliminaries

Problem Formulation. We model robotic manipulation as a Partially Observable Markov Decision Process (POMDP) (Kaelbling et al., 1998), defined as:

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

where $\mathcal{S}$ denotes the state space of the robot and environment, $\mathcal{A}$ is the action space, $\mathcal{O}$ represents the multimodal observation space (*e.g.*, RGB and proprioception), and $\mathcal{L}$ denotes the space of natural-language task instructions. At the start of a task episode, the VLA policy π_θ receives an instruction $l \in \mathcal{L}$ and an initial observation o_0. The goal of the VLA policy π_θ is to generate a sequence of actions:

$$a_{0:T-1} \sim \pi_\theta(a_t \mid o_{t-H+1:t}, l), \tag{2}$$

where o_t and a_t denote the observation and action at time t, T is the episode length, and H is the number of past observations used as policy input.

The above formulation characterizes the standard VLA decision process. However, it inherently assumes a fixed, pre-trained policy. Real-world deployments, on the other hand, demand adaptability to dynamic environments. Therefore, under the test-time adaptation, our goal should now switch to adjusting the pretrained policy π_θ online during deployment flexibly *without* any access to training data, environment resets, or human intervention.

Proximal Policy Optimization (PPO). PPO is an actor–critic policy-gradient method that uses a clipped surrogate objective to constrain policy updates, ensuring stable optimization by keeping the updated policy within a trust region of the previous policy. Formally, the PPO objective is defined as:

$$L^{\text{PPO}}(\theta) = \mathbb{E}_t \left[L_t^{\text{CLIP}}(\theta) - c_1 L_t^{\text{Value}}(\theta) + c_2 L_t^{\text{entropy}}(\theta) \right], \tag{3}$$

where $L_t^{\text{CLIP}}(\theta)$ represents the clipped policy loss, $L_t^{\text{Value}}(\theta)$ denotes the value function loss, $L_t^{\text{Entropy}}(\theta)$ is the entropy regularization term, and c_1 and c_2 are weighting coefficients. The clipped policy objective is defined as:

$$L_t^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min\Big(r_t(\theta)\hat{A}_t,\ \text{clip}\big(r_t(\theta), 1-\epsilon, 1+\epsilon\big)\hat{A}_t \Big) \right], \tag{4}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the ratio between the new and old policy, ϵ controls the clipping range, $\hat{A}_t$ denotes the advantage estimate, and clip$(\cdot)$ denotes the clipping operation. PPO typically employs Generalized Advantage Estimation (GAE) to estimate $\hat{A}_t$, given by:

$$\hat{A}_t = \textstyle\sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}, \tag{5}$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ is temporal-difference (TD) residual, γ is the discount factor, λ is the smoothing parameter, and $V(s_t)$ is the estimated expected return from state s_t.

3.2 TT-VLA

In PPO, both the policy π_θ and value function V_θ are trained jointly. However, in VLA test-time adaptation, learning a reliable value function is generally infeasible due to two reasons: ① ***Limited samples***: Test-time adaptation relies on extremely limited interaction data, a single episode, which is insufficient for accurate return estimation. ② ***Strict time constraints***: Test-time updates for VLA models must be performed online under strict latency constraints. To overcome these limitations, we develop a value-free PPO, which enables policy adaptation *without* an explicit value function.

Dense Progress-Based Reward. Most existing RL-based VLAs (Li et al., 2025a; Liu et al., 2025a) rely on sparse terminal rewards, typically indicating binary task success or failure at the end of an episode. While such rewards are effective during offline training, where episodes can be replayed or reset, they are impractical for test-time adaptation: the policy updates are delayed until task completion, preventing any mid-episode correction or online adjustment.Consequently, a robot operating

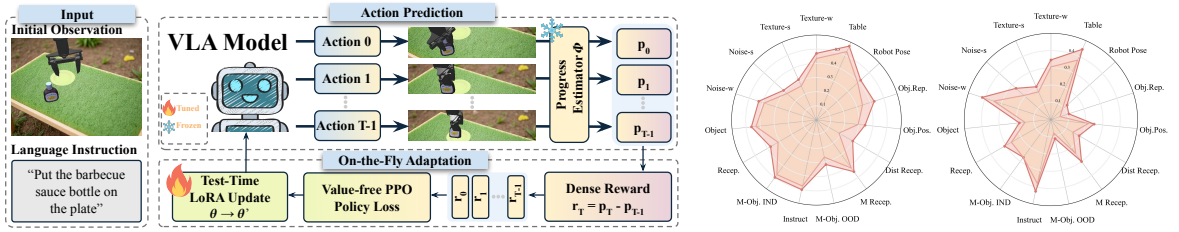


Figure 2: **Overview of TT-VLA. (a) Overall pipeline.** In TT-VLA, a pretrained VLA policy receives an observation and instruction, executes actions in the environment, and receives dense, progress-based rewards computed by a progress estimator. These rewards are used to update the policy online via a value-free PPO objective, enabling continuous within-episode policy adaptation at test time (see §3.2). **(b) Effectiveness.** TT-VLA consistently improves the performance of diverse VLA backbones across unseen tasks, demonstrating robust generalization and adaptability under evolving conditions or distributional shifts (see §4.2-4.3).

with sparse rewards cannot refine its behavior during inference, leading to fragile and non-adaptive performance in dynamic environments.

Let $p_t \in [0, 1]$ denote task progress at time step t. Intuitively, p_t should increase when actions advance task completion and decrease when actions undo or reverse previously achieved progress. In a POMDP setting, we estimate progress directly from observations and language instructions as:

$$p_t = \Phi(o_{0:t+1}, l), \tag{6}$$

where Φ denotes a task progress predictor conditioned on the observation history and instruction l. The per-step dense reward is then defined as the temporal difference in progress:

$$r_t = p_t - p_{t-1}. \tag{7}$$

We instantiate Φ using the Vision-Language-Action-Critic model (VLAC) (Zhai et al., 2025), a pretrained multimodal model that serves as a scalar regressor for task progress estimation.

This progress-based reward exhibits three desirable properties. First, it requires no external supervision during inference, allowing fully autonomous adaptation at test time. Second, it provides dense, step-wise feedback that facilitates continuous mid-episodic policy adaptation. Third, it encourages monotonic progress toward task completion while discouraging oscillatory or regressive behaviors.

Training Objective. Under the test-time VLA setting, learning an accurate value function V_θ within a single episode is impractical. We therefore adopt a value-free PPO variant that removes the value function learning and directly uses the per-step reward signal from Eq. 7 for policy adaptation.

Starting from Eq. 3, we remove auxiliary losses by setting $c_1 = 0$ and $c_2 = 0$, discarding both value regression and entropy regularization term. While the entropy term encourages exploration during training, test-time adaptation prioritizes rapid fitting of the current task rather than broad exploration. As a result, our objective focuses solely on stable policy refinement, while preserving the clipped surrogate loss. Eq. 3 now turns into:

$$L(\theta) = \mathbb{E}_t \left[L_t^{\text{CLIP}}(\theta) \right]. \tag{8}$$

To make the agent precisely capture the immediate value of the current action, we further redefine the advantage to depend only on the reward obtained from that action, rather than the exponentially weighted combination of TD residuals used in GAE (see Eq. 5). In other words, we focus on how each individual action contributes to instantaneous progress rather than estimating long-term returns. To achieve this, we set $\lambda = 0$ and $\gamma = 0$, collapsing GAE into a one-step formulation:

$$\hat{A}_t = \delta_t = r_t. \tag{9}$$

Here, $\delta_t = r_t$ since we remove the value function, making the TD residual the immediate reward signal. This simplification ensures that policy updates directly reflect the progress at each step, allowing the agent to adapt rapidly to changing conditions without relying on a value function.

Overall Pipeline. At the beginning of each episode, the pretrained VLA receives an initial observation o_0 and instruction l. The VLA policy π_θ generates the first action a_0. At each subsequent time step t, the VLA receives the latest observation o_t and outputs an action a_t. After execution, the progress estimator Φ computes the task progress p_t, and the corresponding dense reward r_t (see Eq. 7). This reward is used to compute the policy loss via Eq. 8 in a value-free manner, and the policy parameters θ are updated accordingly. The updated policy $\pi_{\theta'}$ is then used to generate subsequent actions, allowing

continuous refinement throughout the episode.The pseudo code is shown in Appendix Algorithm 1.

3.3 Theoretical Analysis of TT-VLA

In §3.2, we proposed TT-VLA using a progress-based dense reward and a value-free formulation for test-time adaptation. In this section, we provide a theoretical justification for these design choices.

Proposition 1 (Vanishing learning signal under progress-difference reward)**.** *Let the per-step reward be defined as the progress difference* $r_t = p_t - p_{t-1}$ *with* $p_t \in [0, 1]$. *Assume that the value function represents the remaining progress,* $V(s_t) = 1 - p_{t-1}$, *and the discount factor is* $\gamma = 1$. *Then the temporal-difference (TD) error vanishes for all* t, *and consequently the GAE* $\hat{A}_t$ *is identically zero:*

$$\delta_t = 0, \quad \hat{A}_t = 0, \qquad \forall \lambda \in [0, 1]. \tag{10}$$

Proof. Substituting $r_t = p_t - p_{t-1}$, $V(s_t) = 1 - p_{t-1}$, and $\gamma = 1$ into the TD residual yields

$$\begin{aligned} \delta_t &= r_t + \gamma V(s_{t+1}) - V(s_t) \\ &= (p_t - p_{t-1}) + (1 - p_t) - (1 - p_{t-1}) \\ &= 0. \end{aligned} \tag{11}$$

By Eq. 5, GAE is a weighted sum of TD residuals, it follows that $\hat{A}_t = 0$. Therefore, the policy gradient receives no learning signal. ■

Corollary 1 (Negative TD bias when $\gamma < 1$)**.** *Let* $r_t = p_t - p_{t-1}$ *with* $p_t \in [0, 1]$ *and* $V(s_t) = 1 - p_{t-1}$. *If* $0 < \gamma < 1$, *then TD residual* $\delta_t < 0$, *introducing a systematic negative bias in advantage estimation.*

Proof. Substituting $r_t = p_t - p_{t-1}$ and $V(s_t) = 1 - p_{t-1}$ into the TD residual, we get

$$\begin{aligned} \delta_t &= r_t + \gamma V(s_{t+1}) - V(s_t) \\ &= (p_t - p_{t-1}) + \gamma(1 - p_t) - (1 - p_{t-1}) \\ &= (\gamma - 1)(1 - p_t). \end{aligned} \tag{12}$$

Since $0 < \gamma < 1$ and $1 - p_t > 0$, this implies $\gamma - 1 < 0$ and thus $\delta_t < 0$. ■

Lemma 1 (One-step collapse of GAE)**.** *Let the reward be defined as the progress difference*

$$r_t = p_t - p_{t-1}, \tag{13}$$

and let the value function estimator be $V(s)$. *Then, for GAE:*

$$\hat{A}_t = \textstyle\sum_{l=0}^{\infty} (\gamma\lambda)^l \, \delta_{t+l}, \tag{14}$$

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t), \tag{15}$$

the following statements hold:
(a). If $\lambda = 0$, *then* $A_t = \delta_t$.
(b). If $\gamma = 0$, *then* $A_t = \delta_t = r_t - V(s_t)$. *In particular, if* $V(s) \equiv 0$, *then* $A_t = r_t$.

The proof is provided in Appendix §S2.

4 Experiment

In this section, we present a comprehensive evaluation of our proposed method through a series of unseen robotic tasks. We detail the task setups, outline the implementation specifics, and compare our approach against baselines. More experimental details are provided in Appendix §S3-S6.

4.1 Experimental Setup

Environment and Task Settings. As stated in §2.1, TT-VLA aims to address the inherent vulnerability of current VLAs to unanticipated dynamics and domain shifts. To evaluate this generalization capability on ***unseen tasks***, we test TT-VLA on both simulated and real-world tasks.

For ***simulation experiments*** (see §4.2), we follow RL4VLA's (Liu et al., 2025a) setup, focusing on a standard pick-and-place manipulation scenario. The agent receives an RGB observation and a natural-language instruction, and outputs a Cartesian end-effector delta along with a binary gripper command. Specifically, as in (Liu et al., 2025a), we evaluate generalization along three dimensions: *Execution*, *Vision*, and *Semantics*. For *Execution*, the initial poses of the robot, object, and receptacle are randomized, and an additional mid-episode object repositioning condition is introduced to evaluate robustness to dynamic variations during execution. For *Vision*, both foreground and background appearances are varied through dynamic textures, unseen table surfaces, and image-level noise. For *Semantics*, unseen objects, receptacles, and instruction paraphrases are introduced, along with multi-object, multi-receptacle, and distractor-receptacle tasks designed to assess compositional and linguistic generalization. Detailed task specifications are provided in Appendix §S3. All simulation experiments are conducted in ManiSkill 3 (Tao et al., 2024) using a WidowX-250S robotic arm.

For ***real-world evaluation*** (see §4.3), we study pick-and-place manipulation tasks on a Franka Research 3 platform. The agent similarly receives an RGB image and a task instruction, and outputs a Cartesian end-effector displacement together with a binary gripper command. We evaluate performance

Model	*Execution*				*Vision*					
	Obj. Pos.	**Robot Pose**	**Obj. Rep.**	**Avg.**	**Table**	**Texture-w**	**Noise-w**	**Texture-s**	**Noise-s**	**Avg.**
Nora (Hung et al., 2025)	23.75%	10.83%	7.50%	14.03%	39.72%	32.50%	36.67%	19.58%	22.92%	29.92%
Nora + TT-VLA	25.00%	12.50%	10.83%	16.11%	44.58%	34.58%	41.67%	20.83%	27.08%	33.75%
Δ	+1.25%	+1.67%	+3.33%	2.08%	+6.66%	+2.08%	+5.00%	+1.25%	+4.16%	+3.83%
Relative Gain	↑ 5.26%	↑ 15.42%	↑ 44.40%	↑ 14.85%	↑ 17.56%	↑ 6.40%	↑ 13.64%	↑ 6.38%	↑ 18.15%	↑ 12.80%
OpenVLA (Kim et al., 2024)	31.67%	41.25%	36.25%	36.39%	54.85%	45.42%	40.83%	28.33%	30.00%	39.83%
OpenVLA + TT-VLA	34.58%	42.08%	42.92%	39.83%	57.08%	47.08%	42.92%	31.25%	31.33%	41.93%
Δ	+2.92%	+0.83%	+6.67%	+3.45%	+2.50%	+1.67%	+2.08%	+2.92%	+1.33%	+2.10%
Relative Gain	↑ 9.21%	↑ 2.02%	↑ 18.40%	↑ 9.54%	↑ 4.58%	↑ 3.67%	↑ 5.10%	↑ 10.29%	↑ 4.43%	↑ 5.27%
OpenVLA-RL (Liu et al., 2025a)	82.08%	81.25%	81.25%	81.53%	87.08%	85.00%	85.83%	64.17%	69.17%	78.25%
OpenVLA-RL + TT-VLA	84.17%	82.08%	86.25%	84.17%	90.00%	86.25%	85.83%	65.83%	72.08%	80.00%
Δ	+2.09%	+0.83%	+5.00%	+2.64%	+2.92%	+1.25%	+0.00%	+1.66%	+2.91%	+1.75%
Relative Gain	↑ 2.54%	↑ 1.02%	↑ 6.15%	↑ 3.24%	↑ 2.08%	↑ 1.47%	↑ 0.00%	↑ 2.59%	↑ 4.21%	↑ 2.23%
TraceVLA (Zheng et al., 2025)	55.00%	18.75%	7.08%	26.94%	71.67%	67.08%	67.08%	45.83%	45.83%	59.50%
TraceVLA + TT-VLA	57.92%	21.50%	7.50%	28.97%	72.92%	67.50%	67.92%	46.25%	47.08%	60.33%
Δ	+2.92 %	+2.75%	+0.42%	+2.03%	+1.25%	+0.42%	+0.84%	+0.42%	+1.25%	+0.84%
Relative Gain	↑5.31%	↑14.67%	↑5.93%	↑7.53%	↑1.47%	↑0.63%	↑1.25%	↑0.92%	↑2.73%	↑1.41%

Model	*Semantics*							
	M-Obj. OOD	**Instruct**	**Dist Recep.**	**Recep.**	**M Recep.**	**Object**	**M-Obj. IND**	**Avg.**
Nora (Hung et al., 2025)	10.00%	39.85%	16.67%	28.33%	26.25%	17.08%	27.08%	23.57%
Nora + TT-VLA	11.25%	42.50%	18.75%	30.83%	30.00%	18.33%	27.08%	25.53%
Δ	+1.25%	+2.92%	+2.08%	+2.50%	+3.75%	+1.25%	+0.00%	+1.96%
Relative Gain	↑ 12.50%	↑ 7.38%	↑ 12.48%	↑ 8.82%	↑ 14.29%	↑ 7.32%	↑ 0.00%	↑ 8.33%
OpenVLA (Kim et al., 2024)	28.75%	49.58%	20.42%	33.33%	43.75%	45.00%	49.58%	38.63%
OpenVLA + TT-VLA	32.05%	50.17%	29.58%	37.50%	45.00%	46.25%	50.00%	41.51%
Δ	+3.30%	+0.28%	+9.17%	+4.17%	+1.25%	+1.25%	+0.42%	+2.88%
	↑ 11.48%	↑ 1.18%	↑ 44.90%	↑ 12.50%	↑ 2.86%	↑ 2.78%	↑ 0.85%	↑ 7.54%
OpenVLA-RL (Liu et al., 2025a)	62.50%	86.67%	60.00%	79.58%	80.42%	77.50%	77.50%	74.88%
OpenVLA-RL + TT-VLA	65.00%	90.00%	61.25%	79.58%	80.83%	78.33%	80.00%	76.43%
Δ	+2.50%	+3.33%	+1.25%	+0.00%	+0.41%	+0.83%	+2.50%	+1.55%
Relative Gain	↑ 4.00%	↑ 3.84%	↑ 2.08%	↑ 0.00%	↑ 0.51%	↑ 1.07%	↑ 3.23%	↑ 2.06%
TraceVLA (Zheng et al., 2025)	22.50%	59.17%	27.92%	47.50%	55.83%	45.00%	57.92%	45.12%
TraceVLA + TT-VLA	25.00%	60.00%	28.33%	51.25%	55.83%	47.08%	60.00%	46.78%
Δ	+2.50%	+0.83%	+0.41%	+3.75%	+0.00%	+2.08%	+2.08%	+1.66%
Relative Gain	↑11.11%	↑1.40%	↑1.47%	↑7.89%	↑0.00%	↑4.62%	↑3.59%	↑3.69%

Table 1: **Main results on unseen simulation tasks.** We report success rates (%) across three generalization dimensions: *Execution*, *Vision*, and *Semantics* on 4 state-of-the-art open-source VLAs (*i.e.*, Nora, OpenVLA, OpenVLA-RL, and TraceVLA). Δ denotes the absolute improvement, and ↑ indicates relative gains. As shown in the table, across all baselines and task categories, TT-VLA consistently improves performance during test time, demonstrating substantial generalizability and constituting a novel angle for VLA adaptivity.

on nine unseen tasks designed to assess robustness to executional, visual, and semantic shifts.

Implementation Details. For simulation, each task is executed for 80 trials across three random seeds using a 640×480 RGB image as input. For real-world experiments, each task is evaluated over 10 trials under consistent experimental conditions, including fixed camera viewpoints with an image resolution of 500×480, controlled lighting, and static backgrounds. The policy is fine-tuned using LoRA (Hu et al., 2022) with ranks $\{16, 32\}$. Learning rates are chosen from $\{1 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}\}$, optimized with AdamW. A clip parameter ϵ of 0.2 is applied to enhance training stability. Each episode is executed with a fixed horizon of 160 steps.

4.2 Simulation Results

Baselines. We benchmark our proposed method against several state-of-the-art open-source VLA models, which span diverse architectural designs and training paradigms:

- **Nora** (Hung et al., 2025) adopts Qwen-2.5-VL-3B (Bai et al., 2025a) as its backbone and employs the FAST+ tokenizer (Pertsch et al., 2025) to enable efficient action sequence generation. Following (Liu et al., 2025a) to ensure a strong initial policy, we further fine-tune Nora for 50k steps on a self-collected ManiSkill 3 dataset.
- **OpenVLA** (Kim et al., 2024) is one of the most widely used open-source VLA models. It is built on Llama-2-7B (Touvron et al., 2023). Consistent with (Liu et al., 2025a), we apply a warm-up fine-tuning phase of 10k steps prior to evaluation.
- **OpenVLA-RL** (Liu et al., 2025a) extends OpenVLA via reinforcement learning during training, enabling further task-specific policy refinement beyond supervised pre-training.
- **TraceVLA** (Zheng et al., 2025) is designed to enhance spatio-temporal reasoning through visual trace prompting. By encoding state–action histories as visual prompts, it better captures long-horizon dependencies and improves manipulation performance in interactive environments.

Overall Performance. As shown in Table 1, our proposed method **consistently improves** the perfor-

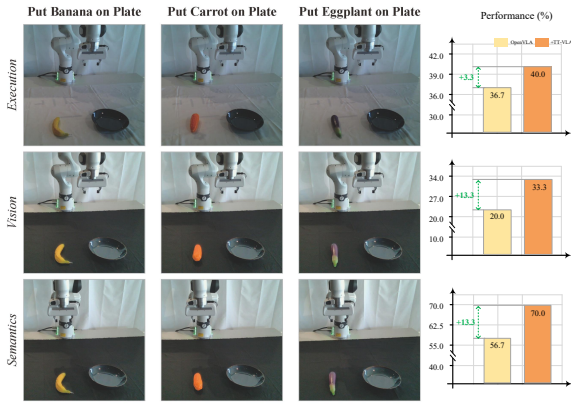


Figure 3: **Real-world setup and evaluation.** We evaluate nine real-world pick-and-place tasks covering *Execution*, *Vision*, and *Semantics* generalization, with three tasks per category. The results show that TT-VLA consistently improves performance over baseline VLA models in real-world settings.

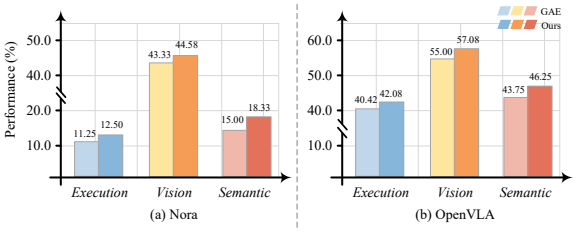


Figure 4: **Impact of reward design**. The results show that our progress-based reward consistently outperforms the standard GAE across tasks and models.

mance of **all** representative baseline models across a range of unseen tasks, demonstrating its broad applicability. For example, when applied to Nora, our method achieves gains on **14 out of 15** tasks, with relative improvements ranging from **5.26% to 44.4%**. The largest improvements are observed on Task Obj. Rep. (44.4%) and Task Noise-s (18.15%). Similarly, on OpenVLA, our method yields consistent performance improvements across nearly all tasks, including several large-margin gains (*e.g.*, 44.9% on Task Disturb Recep. and 18.4% on Task Obj. Rep.). Overall, while current methods generally focus on training-time sophisticated architectural improvements, we demonstrate that substantial generalizability across diverse baselines and unseen tasks can be achieved through a markedly more streamlined yet robust test-time adjustment. Given its capacity for dynamic adjustment based on unseen samples, our approach constitutes a significantly novel solution for VLA adaptivity (Kachaev et al., 2025; Liu et al., 2025a).

4.3 Real-World Results

We use OpenVLA as the base policy, and evaluate TT-VLA on nine unseen real-world tasks (see Fig. 3). As seen, our method consistently improves performance over OpenVLA in real-world settings, demonstrating effective generalization beyond simulation and highlighting the practicality of test-time adaptation in real robotic deployments.

We further report a representative case study "put banana on plate" in Fig. 5, where the robot grasps the banana and moves it toward the plate. During the original execution, the gripper temporarily deviates from the target region and moves away from the plate, signifying a substantial risk of task failure. However, by leveraging the dense, progress-based reward of TT-VLA, the policy enables rapid detection of task regression and on-the-fly behavioral adjustments. The immediate reward feedback allows the VLA policy to correct deviations and realign with the task objective, ultimately completing the placement successfully. This example highlights the advantage of instantaneous, progress-aware rewards in enabling rapid recovery from execution errors. More real-world qualitative examples are shown in Appendix §S6.

4.4 Diagnostic Experiments

We conduct an ablation study on both Nora and OpenVLA using three representative unseen tasks.

Reward/Advantage Design. We first analyze the effect of discounting in GAE (see Eq. 5). Specifically, we compare the standard GAE setting with nonzero discount factor and trace parameter (*e.g.*, $\gamma > 0, \lambda > 0$) against our variant in which both γ and λ are set to zero. By eliminating discounting and trace accumulation, TT-VLA emphasizes how each individual action contributes to immediate progress rather than estimating long-term returns.

Empirically, focusing on immediate progress yields consistent improvements in performance. For example, as shown in Fig. 4, on the *Vision* task with OpenVLA, our setting achieves a success rate of 57.08%, compared to 55.00% when using standard GAE. We attribute this performance gain to the fact that long-horizon returns can be unreliable in this setting, occasionally assigning overly optimistic advantage signals to ineffective

Learning Step	*Execution*		*Vision*		*Semantics*	
	Nora	OpenVLA	Nora	OpenVLA	Nora	OpenVLA
1	10.83%	40.42%	40.00%	54.12%	15.42%	43.33%
4	11.25%	41.25%	42.08%	56.25%	17.50%	**46.25%**
8	**12.50%**	**42.08%**	**44.58%**	**57.08%**	**18.33%**	**46.25%**
16	11.25%	**42.08%**	43.33%	55.42%	17.50%	45.42%

Table 2: **Impact of learning step**. An update interval of 8 steps yield the best performance.

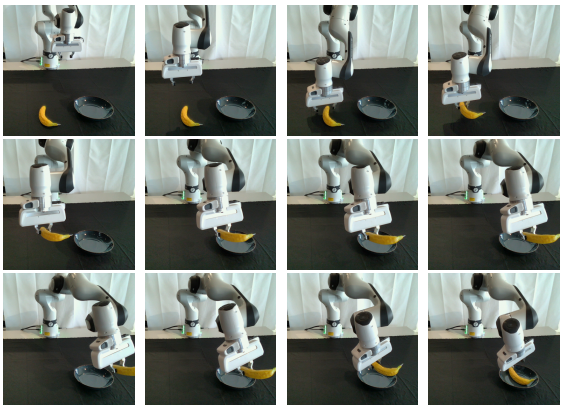


Figure 5: **Real-world case study** illustrates how TT-VLA's instantaneous reward feedback enables rapid recovery from trajectory errors during deployment.

actions. These results suggest that instantaneous feedback can be more effective than incorporating discounted future rewards during test time.

Test-Time Training Steps. We then explore the impact of model update frequency in TT-VLA by varying the update interval over different environment steps (*i.e.*, 1, 4, 8, and 16). The number of learning steps is designed to balance the trade-off between rapid adaptation to newly collected data and the overall stability of the optimization process. Table 2 shows that updating the model every 8 steps yields the best performance. More frequent updates (*e.g.*, 1 step) can destabilize training and limit the effectiveness of each update. In contrast, less frequent updates (*e.g.*, 16 steps) delay policy improvement and reduce learning efficiency. These findings suggest that an intermediate update frequency achieves a balance between rapid policy adaptation and stable optimization. Additional details are provided in Appendix §S4.

4.5 Discussions on VLA Test-Time Training

As stated in §2.2, TTT was originally proposed for LLMs. A natural question is: *Can test-time training methods in LLMs be directly applied to VLA models?* To address this question, we examine two representative approaches for VLAs: a self-supervised test-time training method, TLM (Hu et al., 2025), and a test-time reinforcement learning method, TTRL (Hu et al., 2025). Unless otherwise specified, we follow the same experimental setup as the diagnostic experiments, using the same tasks and baseline models for evaluation.

We first consider TLM (Hu et al., 2025) that enables test-time adaptation by directly minimizing input perplexity without any external supervision. When applied to VLAs, TLM updates model parameters by optimizing the likelihood of test-time observation sequences. As shown in Table 3, this strategy results in lower performance gains than TT-VLA. The reason is that, unlike pure language tasks, VLA tasks involve complex interactions between perceptions, instruction understanding, and actions. Updates driven solely by input likelihood may overly emphasize representational consistency rather than task-oriented decision making. As a result, self-supervised test-time objectives cannot readily translate to the VLA domain.

Model	*Execution*		*Vision*		*Semantics*	
	Nora	OpenVLA	Nora	OpenVLA	Nora	OpenVLA
TLM	11.25%	40.42%	41.25%	52.50%	16.67%	42.9%
TTRL	10.42%	40.83%	39.58%	51.42%	16.25%	41.76%
Ours	**12.50%**	**42.08%**	**44.58%**	**57.08%**	**18.33%**	**46.25%**

Table 3: **Comparison of TTT methods**. We compare TT-VLA with TLM and TTRL (see §4.5). As seen, TT-VLA achieves superior performance, highlighting the importance of progress-based reward for effective test-test adaptation in VLAs.

We further compare TT-VLA with TTRL (Hu et al., 2025), a recently proposed test-time reinforcement learning approach. TTRL performs test-time adaptation by sampling multiple candidate outputs for each input and constructing a consensus pseudo-label via majority voting (Shao et al., 2024). This pseudo-label is then used to construct rule-based rewards, where outputs that match/mismatch the pseudo-label receive positive/zero rewards. As reported in Table 3, TTRL underperforms our proposed TT-VLA, indicating that the consensus-based pseudo-label is less effective for VLA tasks. One possible reason is that majority voting does not reflect action quality, and reward signals derived from output agreement fail to provide task-aligned learning signals, thereby limiting the effectiveness of VLA test-time updates. More details of TLM and TTRL are provided in Appendix §S5.

5 Conclusion

While VLA models have gained significant popularity on closed-form benchmarks, this work focuses on the flexibility of applying these models in evolving environments via test-time reinforcement learning. Experimental results demonstrate that TT-VLA consistently enhances performance on unseen tasks across diverse simulated and real-world scenarios, as well as across various VLA backbones. We believe that our framework provides pioneering and foundational contributions to VLA models to flexibly refine action policies under dynamic, previously unseen test-time cases.

Limitations

Although TT-VLA supports on-the-fly adaptation during deployment, its effectiveness is subject to several factors. First, TT-VLA relies on task-progress-based reward signals at test time. In scenarios involving highly ambiguous visual states, severe occlusions, or tasks that do not exhibit monotonic progress, imperfect progress estimates may introduce noisy reward signals, potentially limiting performance gains. In addition, TT-VLA builds upon a pretrained base VLA model. When the base policy is weak or poorly aligned with the unseen task, TT-VLA may yield limited improvements, which is a well-recognized yet persistent bottleneck in the VLA generalization (Li et al., 2025a; Liu et al., 2025a).

Ethical Considerations

We conform to the ACL Code of Ethics and further show the consent to our work below. All datasets used in this study are publicly available and released under permissive licenses, and all the models are publicly available (see Appendix §S10 for Asset License and Consent). We would like to state that the contents in the dataset do NOT represent our views or opinions, and our paper does not involve crowdsourcing or research with human subjects.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*.

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B Tenenbaum, Tommi S Jaakkola, and Pulkit Agrawal. 2023. Is conditional generative modeling all you need for decision making? In *ICLR*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025a. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

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

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, and 32 others. 2023. RT-1: robotics transformer for real-world control at scale. In *RSS*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. In *NeurIPS*.

Hardy Chen, Haoqin Tu, Fali Wang, Hui Liu, Xianfeng Tang, Xinya Du, Yuyin Zhou, and Cihang Xie. 2025a. Sft or rl? an early investigation into training r1-like reasoning large vision-language models. *arXiv preprint arXiv:2504.11468*.

Yuhui Chen, Shuai Tian, Shugao Liu, Yingting Zhou, Haoran Li, and Dongbin Zhao. 2025b. Conrft: A reinforced fine-tuning method for vla models via consistency policy. In *RSS*.

Yuxuan Chen and Xiao Li. 2025. Rlrc: Reinforcement learning-based recovery for compressed vision-language-action models. *arXiv preprint arXiv:2506.17639*.

Zengjue Chen, Runliang Niu, He Kong, and Qi Wang. 2025c. Tgrpo: Fine-tuning vision-language-action model via trajectory-wise group relative policy optimization. *arXiv preprint arXiv:2506.08440*.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. 2023. Diffusion policy: Visuomotor policy learning via action diffusion. In *RSS*.

Yiming Cui, Cheng Han, and Dongfang Liu. 2024. Collaborative multi-task learning for multi-object tracking and segmentation. *Journal on Autonomous Transportation Systems*.

Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. 2021. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*.

Senyu Fei, Siyin Wang, Li Ji, Ao Li, Shiduo Zhang, Liming Liu, Jinlong Hou, Jingjing Gong, Xianzhong Zhao, and Xipeng Qiu. 2025. Srpo: Self-referential policy optimization for vision-language-action models. *arXiv preprint arXiv:2511.15605*.

Sachin Goyal, Ananya Kumar, Sankalp Garg, Zico Kolter, and Aditi Raghunathan. 2023. Finetune like you pretrain: Improved finetuning of zero-shot vision models. In *CVPR*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025a. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*.

Haiyang Guo, Fanhu Zeng, Fei Zhu, Jiayi Wang, Xukai Wang, Jingang Zhou, Hongbo Zhao, Wenzhuo Liu, Shijie Ma, Xu-Yao Zhang, and 1 others. 2025b. A comprehensive survey on continual learning in generative models. *arXiv preprint arXiv:2506.13045*.

Yanjiang Guo, Jianke Zhang, Xiaoyu Chen, Xiang Ji, Yen-Jen Wang, Yucheng Hu, and Jianyu Chen. 2025c. Improving vision-language-action model with online reinforcement learning. *arXiv preprint arXiv:2501.16664*.

Cheng Han, Qifan Wang, Yiming Cui, Zhiwen Cao, Wenguan Wang, Siyuan Qi, and Dongfang Liu. 2023. E^2vpt: An effective and efficient approach for visual prompt tuning. In *ICCV*.

Jisu Han, Jihee Park, Dongyoon Han, and Wonjun Hwang. 2025. When test-time adaptation meets self-supervised models. *arXiv preprint arXiv:2506.23529*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. In *ICLR*.

Jinwu Hu, Zhitian Zhang, Guohao Chen, Xutao Wen, Chao Shuai, Wei Luo, Bin Xiao, Yuanqing Li, and Mingkui Tan. 2025. Test-time learning for large language models. *arXiv preprint arXiv:2505.20633*.

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

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

Jiangyong Huang, Silong Yong, Xiaojian Ma, Xiongkun Linghu, Puhao Li, Yan Wang, Qing Li, Song-Chun Zhu, Baoxiong Jia, and Siyuan Huang. 2023. An embodied generalist agent in 3d world. *arXiv preprint arXiv:2311.12871*.

Chia-Yu Hung, Qi Sun, Pengfei Hong, Amir Zadeh, Chuan Li, U Tan, Navonil Majumder, Soujanya Poria, and 1 others. 2025. Nora: A small open-sourced generalist vision language action model for embodied tasks. *arXiv preprint arXiv:2504.19854*.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. 2022. Planning with diffusion for flexible behavior synthesis. In *ICML*.

Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. 2022. Visual prompt tuning. In *ECCV*.

Anqing Jiang, Yu Gao, Yiru Wang, Zhigang Sun, Shuo Wang, Yuwen Heng, Hao Sun, Shichen Tang, Lijuan Zhu, Jinhao Chai, and 1 others. 2025a. Irl-vla: Training an vision-language-action policy via reward world model. *arXiv preprint arXiv:2508.06571*.

Bo Jiang, Shaoyu Chen, Qian Zhang, Wenyu Liu, and Xinggang Wang. 2025b. Alphadrive: Unleashing the power of vlms in autonomous driving via reinforcement learning and reasoning. *arXiv preprint arXiv:2503.07608*.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. 2022. VIMA: general robot manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*.

Nikita Kachaev, Mikhail Kolosov, Daniil Zelezetsky, Alexey K Kovalev, and Aleksandr I Panov. 2025. Don't blind your vla: Aligning visual representations for ood generalization. *arXiv preprint arXiv:2510.25616*.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence*.

Adilbek Karmanov, Dayan Guan, Shijian Lu, Abdulmotaleb El Saddik, and Eric Xing. 2024. Efficient test-time adaptation of vision-language models. In *CVPR*.

Dongyoung Kim, Sumin Park, Huiwon Jang, Jinwoo Shin, Jaehyung Kim, and Younggyo Seo. 2025. Robot-r1: Reinforcement learning for enhanced embodied reasoning in robotics. *arXiv preprint arXiv:2506.00070*.

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

Petar Kormushev, Sylvain Calinon, and Darwin G Caldwell. 2013. Reinforcement learning in robotics: Applications and real-world challenges. *Robotics*.

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

Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, and 1 others. 2025a. Simplevla-rl: Scaling vla training via reinforcement learning. *arXiv preprint arXiv:2509.09674*.

Hengtao Li, Pengxiang Ding, Runze Suo, Yihao Wang, Zirui Ge, Dongyuan Zang, Kexian Yu, Mingyang Sun, Hongyin Zhang, Donglin Wang, and 1 others.

2025b. Vla-rft: Vision-language-action reinforcement fine-tuning with verified rewards in world simulators. *arXiv preprint arXiv:2510.00406*.

Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, and 1 others. 2023. Vision-language foundation models as effective robot imitators. *arXiv preprint arXiv:2311.01378*.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, and 1 others. 2025c. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*.

Zhixuan Liang, Yao Mu, Mingyu Ding, Fei Ni, Masayoshi Tomizuka, and Ping Luo. 2023. Adaptdiffuser: Diffusion models as adaptive self-evolving planners. *arXiv preprint arXiv:2302.01877*.

Jiaming Liu, Mengzhen Liu, Zhenyu Wang, Pengju An, Xiaoqi Li, Kaichen Zhou, Senqiao Yang, Renrui Zhang, Yandong Guo, and Shanghang Zhang. 2024a. Robomamba: Efficient vision-language-action model for robotic reasoning and manipulation. In *NeurIPS*.

Jijia Liu, Feng Gao, Bingwen Wei, Xinlei Chen, Qingmin Liao, Yi Wu, Chao Yu, and Yu Wang. 2025a. What can rl bring to vla generalization? an empirical study. *arXiv preprint arXiv:2505.19789*.

Yiyang Liu, James Chenhao Liang, Ruixiang Tang, Yugyung Lee, Majid Rabbani, Sohail A. Dianat, Raghuveer Rao, Lifu Huang, Dongfang Liu, Qifan Wang, and Cheng Han. 2025b. Re-imagining multimodal instruction tuning: A representation view. In *ICLR*.

Yuejiang Liu, Parth Kothari, Bastien van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. 2021. TTT++: when does self-supervised test-time training fail or thrive? In *NeurIPS*.

Zequan Liu, Jiawen Lyn, Wei Zhu, Xing Tian, and Yvette Graham. 2024b. Alora: Allocating low-rank adaptation for fine-tuning large language models. In *NAACL*.

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

Xiaosong Ma, Jie Zhang, Song Guo, and Wenchao Xu. 2023. Swapprompt: Test-time prompt adaptation for vision-language models. In *NeurIPS*.

Max Sobol Mark, Tian Gao, Georgia Gabriela Sampaio, Mohan Kumar Srirama, Archit Sharma, Chelsea Finn, and Aviral Kumar. 2024. Policy agnostic rl: Offline rl and online rl fine-tuning of any class and backbone. *arXiv preprint arXiv:2412.06685*.

Oier Mees, Lukás Hermann, and Wolfram Burgard. 2022. What matters in language conditioned robotic imitation learning over unstructured data. *IEEE Robotics and Automation Letters*.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2021. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. In *ICLR*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*.

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

Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Bäck. 2024. Reasoning with large language models, a survey. *CoRR*.

Vitchyr Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. 2020. Skew-fit: State-covering self-supervised reinforcement learning. In *ICML*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. 2025. Rl's razor: Why online reinforcement learning forgets less. *arXiv preprint arXiv:2509.04259*.

Junyang Shu, Zhiwei Lin, and Yongtao Wang. 2025. Rftf: Reinforcement fine-tuning for embodied agents with temporal feedback. *arXiv preprint arXiv:2505.19767*.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. 2020. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*.

Richard S Sutton, Andrew G Barto, and 1 others. 1999a. Reinforcement learning. *Journal of Cognitive Neuroscience*.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999b. Policy gradient methods for reinforcement learning with function approximation. In *NeurIPS*.

Milind Tambe, W Lewis Johnson, Randolph M Jones, Frank Koss, John E Laird, Paul S Rosenbloom, and Karl Schwamb. 1995. Intelligent agents for interactive simulation environments. *AI Magazine*.

Shuhan Tan, Kairan Dou, Yue Zhao, and Philipp Krähenbühl. 2025. Interactive post-training for vision-language-action models. *arXiv preprint arXiv:2505.17016*.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-kai Chan, and 1 others. 2024. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *arXiv preprint arXiv:2410.00425*.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. 2024. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Luong Trung, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. Reft: Reasoning with reinforced fine-tuning. In *ACL*.

Taowen Wang, Dongfang Liu, James Chenhao Liang, Wenhao Yang, Qifan Wang, Cheng Han, Jiebo Luo, and Ruixiang Tang. 2025a. Exploring the adversarial vulnerabilities of vision-language-action models in robotics. In *ICCV*.

Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, and 1 others. 2025b. Vla-adapter: An effective paradigm for tiny-scale vision-language-action model. *arXiv preprint arXiv:2509.09372*.

Yinjie Wang, Ling Yang, Bowen Li, Ye Tian, Ke Shen, and Mengdi Wang. 2025c. Revolutionizing reinforcement learning framework for diffusion large language models. *arXiv preprint arXiv:2509.06949*.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. In *ICLR*.

Junjie Wen, Minjie Zhu, Jiaming Liu, Zhiyuan Liu, Yicun Yang, Linfeng Zhang, Shanghang Zhang, Yichen Zhu, and Yi Xu. 2025a. dvla: Diffusion vision-language-action model with multimodal chain-of-thought. *arXiv preprint arXiv:2509.25681*.

Junjie Wen, Yichen Zhu, Minjie Zhu, Zhibin Tang, Jinming Li, Zhongyi Zhou, Xiaoyu Liu, Chaomin Shen, Yaxin Peng, and Feifei Feng. 2025b. Diffusionvla: Scaling robot foundation models via unified diffusion and autoregression. In *ICML*.

Ruihan Wu, Chuan Guo, Yi Su, and Kilian Q Weinberger. 2021. Online adaptation to label distribution shift. In *NeurIPS*.

Zhou Xian and Nikolaos Gkanatsios. 2023. Chained-diffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation. In *PMLR*.

Zehao Xiao, Shilin Yan, Jack Hong, Jiayin Cai, Xiaolong Jiang, Yao Hu, Jiayi Shen, Qi Wang, and Cees GM Snoek. 2025. Dynaprompt: Dynamic test-time prompt tuning. *arXiv preprint arXiv:2501.16404*.

Charles Xu, Qiyang Li, Jianlan Luo, and Sergey Levine. 2024. Rldg: Robotic generalist policy distillation via reinforcement learning. *arXiv preprint arXiv:2412.09858*.

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

Hee Suk Yoon, Eunseop Yoon, Joshua Tian Jin Tee, Mark A Hasegawa-Johnson, Yingzhen Li, and Chang D Yoo. 2024. C-tpt: Calibrated test-time prompt tuning for vision-language models via text feature dispersion. In *ICLR*.

Runjia Zeng, Cheng Han, Qifan Wang, Chunshu Wu, Tong Geng, Lifu Huangg, Ying Nian Wu, and Dongfang Liu. 2024. Visual fourier prompt tuning. In *NeurIPS*.

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

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

Zijian Zhang, Kaiyuan Zheng, Zhaorun Chen, Joel Jang, Yi Li, Siwei Han, Chaoqi Wang, Mingyu Ding, Dieter Fox, and Huaxiu Yao. 2024. Grape: Generalizing robot policy via preference alignment. *arXiv preprint arXiv:2411.19309*.

Hao Zhao, Yuejiang Liu, Alexandre Alahi, and Tao Lin. 2023. On pitfalls of test-time adaptation. In *ICML*.

Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. 2025. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. In *ICLR*.

Yuhan Zhu, Guozhen Zhang, Chen Xu, Haocheng Shen, Xiaoxin Chen, Gangshan Wu, and Limin Wang. 2024. Efficient test-time prompt tuning for vision-language models. *arXiv preprint arXiv:2408.05775*.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong T. Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, and 35 others. 2023. RT-2: vision-language-action models transfer web knowledge to robotic control. In *CoRL*.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, and 1 others. 2025. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*.

SUMMARY OF THE APPENDIX

This appendix contains additional experimental results and discussions of our work, organized as:

- §S1 provides **more related works** on VLA models
- §S2 provides **Lemma proof**.
- §S3 includes **more details on tasks**.
- §S4 supplies additional information on **diagnostic experiments**.
- §S5 supplies additional discussions on **Test-Time Training**.
- §S6 provides **more qualitative results**.
- §S7 adds discussions on the practicalness of using **Test-Time GRPO in VLAs**.
- §S8 includes the **reproducibility statement and pseudo code** of our method.
- §S9 highlights the **technical contributions** of our method.
- §S10 offers a **summary of licenses and consent**, and lists usage terms for all models and datasets.
- §S11 includes additional discussions on **ethics concerns**.
- §S12 discusses on **future directions**, highlighting potential areas for further research.
- §S13 provides an **AI disclosure**, and notes that AI assistance was limited to grammar checking.

S1 More Related Works

S1.1 More Discussions on VLA

Recent studies (Brohan et al., 2023; Mees et al., 2022; Pong et al., 2020) have advanced the potential of large-scale Vision Language Models (VLMs) as key enablers for generalist robots, demonstrating promising generalization across a variety of scenes (Zitkovich et al., 2023; Jiang et al., 2022; Team et al., 2024; Huang et al., 2023; Li et al., 2023; Cui et al., 2024). They generally achieve action planning via two main branches: I. Discretization-based approaches (Kim et al., 2024; Brohan et al., 2023; Zitkovich et al., 2023), which discretize the action space into a small set of action tokens; and II. Diffusion-based approaches (Chi et al., 2023; Xian and Gkanatsios, 2023; Janner et al., 2022; Liang et al., 2023; Ajay et al., 2023), which integrate diffusion heads for action prediction.

In our study, we focus on and generalize discretization-based approaches. The reason is that most diffusion-head VLA models adopt a separate action decoder, typically a latent diffusion process that maps visual and instruction embeddings to an action embedding stream, followed by a shallow MLP to regress the robot's joint space (Wen et al., 2025b). This design renders reinforcement-learning (RL) optimization impractical (*i.e.*, also for diffusion large language model (DLLM) RL optimization (Wang et al., 2025c)) for three technical reasons: (i) the resulting policy is implicit and does not expose a tractable per-step log-likelihood ($\log \pi_\theta(a \mid s)$), precluding policy-gradient methods (*e.g.*, REINFORCE (Sutton et al., 1999b)/PPO (Schulman et al., 2017)) and entropy regularization; (ii) action emission requires tens of denoising iterations per control step, creating an inner stochastic chain misaligned with environment time, which breaks step-wise credit assignment; and (iii) the diffusion noise-prediction objective is mismatched with return-based RL objectives, while the terminal MLP head is effectively deterministic, suppressing exploration. However, we notice a very recent paper dVLA (Wen et al., 2025a) decodes actions as a discrete, autoregressive token sequence conditioned on state/instruction, making the current RL attempts possible to apply to diffusion-based approaches. While the code is not publicly available for us to evaluate, we highlight that our method can be naturally applied to this line of research.

S1.2 RL Methods for VLA

As we mentioned in our study, recently, some efforts have attempted to apply RL to the VLA training stage, leaving the test-time adjustments underexplored. In light of this view, we aim to fill the last puzzle of on-the-fly policy adaptation. We list some research with high impact on the integration of RL on VLAs.

GRAPE (Zhang et al., 2024) uses Direct Preference Optimization (DPO) (Rafailov et al., 2023) to train VLAs by integrating human preferences. ConRFT (Chen et al., 2025b) introduces Reinforced Fine-Tuning (Trung et al., 2024) to train VLAs in real-world environments, iteratively training VLAs through alternating RL and SFT rounds. ReinboT (Zhang et al., 2025) rises dense reward design and optimized VLA through reward maximization. iRe-VLA (Guo et al., 2025c) proposed an itera-

tive training framework that combines SFT and RL stages to address training instability and computational overhead. RIPT-VLA (Tan et al., 2025) employs REINFORCE Leave-One-Out (RLOO) (Ahmadian et al., 2024) for VLA training. (Liu et al., 2025a) investigates RL's impact on VLA generalization capabilities, demonstrating improvements over SFT in unseen environments, objects, and textures. VLA-RL (Lu et al., 2025) applies the PPO; TGRPO and SimpleVLA-RL (Chen et al., 2025c; Li et al., 2025a) evaluate trajectories and optimize VLA with GRPO variants; RFTF (Shu et al., 2025) uses value models to generate dense rewards in embodied scenarios for VLA online RL; and SRPO (Fei et al., 2025) leverages a world model to generate progress-based dense rewards. Though promising, it is important to note that current RL-based approaches all operate during training, while real-world deployments inevitably involve evolving conditions and distributional shifts at test time, necessitating VLAs capable of adaptive adjustment in response. The approach most relevant to our work is EVOLVE-VLA (Bai et al., 2025b), which utilizes task progress as a reward signal for reinforcement learning. However, we should notice that EVOLVE-VLA optimizes the policy using GRPO, which incurs substantial computational overhead and is therefore less suitable for real-time robotic deployment. This limitation becomes particularly pronounced in real-world robotic settings, where strict latency constraints are critical.

S2 Lemma Proof

In this section, we provide the proof of Lemma 1, which characterizes the relationship between GAE and the reward-only advantage used in TT-VLA. This result formally justifies our value-free test-time optimization objective.

Proof. (i) For (a): When $\lambda = 0$, the geometric weighting term $(\gamma\lambda)^l$ vanishes for all $l > 0$, Utilizing (14), it yields $A_t = \delta_t$.
(ii) For (b): When $\gamma = 0$, (14) and (15) respectively yields

$$A_t = \delta_t,\ \delta_t = r_t - V(s_t), \tag{16}$$

(16) implies that when $V(s) \equiv 0$, there holds

$$A_t = \delta_t = r_t, \tag{17}$$

which completes the proof. ■

S3 Task Details

For simulation tasks, we follow (Liu et al., 2025a) to define three dimensions of generalization problems for unseen tasks, which are *Execution*, *Vision*, and *Semantics*.

The training task setting: At the beginning of each episode, an object is sampled from the 16 training objects and a table appearance is sampled from the 16 training textures. The object and the receptacle (yellow plate) are placed on the table, with their positions uniformly randomized within a rectangular region. The language instruction follows the template "put O on R", where O and R denote the object and receptacle names, respectively.
Execution explores changes in the initial positions of both the object, the receptacle, the robot initial pose, and mid-episode changes in the object's position during task execution.

- Unseen Object & Receptacle Position (Obj. Pos.): The object and the receptacle are placed on the table, with their positions randomized within a larger square region that surrounds the original rectangular area. All other settings follow the Training setting.

- Unseen Robot Init Pose (Robot Pose): At the start of each episode, the initial poses of all robots are randomized instead of being fixed as in the Training setting. All other settings remain identical to the Training setting.

- Mid-Episode Object Reposition (Obj. Rep.): At the fifth timestep of each episode, the object is teleported to a new randomly sampled position on the table. All other settings remain identical to the Training setting.

Vision includes both foreground and background changes, as well as image-level Dynamic Noise, applied with either weak or strong intensity.

- Unseen Table (Table): The table appearance is sampled from 5 unseen appearance.

- Weak Dynamic Texture (Texture-w): In addition to sampling an object and a table appearance, a texture is selected from the 16 available textures at the start of each episode. This texture is cropped and resized at each timestep differently, and overlaid onto the object, receptacle, and robot arm with a transparency factor of 0.3.

- Strong Dynamic Texture (Texture-s): The settings matches the Weak Dynamic Texture setting, except that the image mixing transparency is increased to 0.5.

- Weak Dynamic Noise (Noise-w): In addition to sampling an object and a table appearance, a texture is selected from the 16 available textures at the start of each episode. The texture is cropped and resized at each timestep differently and overlaid over the entire image with a transparency factor of 0.3.

- Strong Dynamic Noise (Noise-s): The settings matches the Weak Dynamic Noise setting, except that the image mixing transparency is increased to 0.5

Semantics considers previously unseen variations in Objects, Receptacles, and Instruction Phrasings.

- Unseen Objects (Object): The object is sampled from 9 unseen objects, while all other settings follow the Training setting.

- Unseen Receptacles (Recep.): In addition to sampling an object and a table appearance, a receptacle is selected from 16 unseen receptacles at the start of each episode, replacing the default training receptacle (yellow plate). All other settings follow the Training setting

- Unseen Instruction Phrasing (Instruct): In addition to sampling an object and a table appearance, a language instruction template is selected from 16 unseen templates (Same as (Liu et al., 2025a)) at the start of each episode, replacing the default instruction ("put O on R"). All other settings follow the Training setting.

- Seen Multi-Object (M-Obj. (IND)): At the beginning of each episode, two distinct objects are sampled from the 16 training objects along with one of the 16 training table appearances. Both objects and the receptacle (yellow plate) are placed on the table, with their positions randomly initialized within a rectangular region.

- Unseen Multi-Object (M-Obj. (OOD)): Two distinct objects are sampled from the nine unseen objects, with all other settings identical to the Seen Multi-Object settings.

- Distractive Receptacle (Dist Recep.): In addition to sampling an object and a table appearance, a distractor receptacle is selected from 16 unseen receptacles at the start of each episode and placed on the table without being used in the task. All other settings follow the Training setting.

- Multi-Receptacle (M Recep.): At the beginning of each episode, an object is sampled from the 16 training objects, two distinct receptacles are sampled from the 16 unseen receptacles, and a table appearance is selected from the 16 training textures. The object and both receptacles are placed on the table, with their positions randomly initialized within a rectangular region.

For real-world evaluation, we assess our method on nine unseen manipulation tasks designed to test generalization across execution, vision, and semantic dimensions. The execution tasks consist of are "put banana on plate", "put lemon on plate", "put apple on plate" under different initial robot configurations, evaluating robustness to variations in starting states. The vision tasks use the same instructions but introduce different background appearances to assess visual generalization. The semantic tasks also follow the same instruction templates but involve an unseen plate at test time, evaluating the model's ability to generalize to novel semantic contexts. The nine tasks are illustrated in Fig. 3.

S4 Additional Details on Diagnostic Experiments

This section provides additional implementation details for the diagnostic experiments discussed in §4.4. We conduct diagnostic experiments using Nora and OpenVLA. We evaluate one task from each dimension: *execution*, *vision*, and *semantics*. Specifically, we use Task Robot Pose (*execution*), Task Table (*vision*), and Task Object (*semantics*) for all ablations to ensure controlled and comparable evaluations across settings.

For Advantage Design, the standard GAE baseline uses a discount factor of $\gamma = 0.99$ and a trace parameter of $\lambda = 0.95$, with a truncated horizon length of $l = 8$ for advantage estimation. In contrast, our method disables both discounting and trace accumulation by setting $\gamma = 0$ and $\lambda = 0$, yielding a one-step, reward-only advantage.

S5 Additional Details on Test-Time Training Discussions

This section provides implementation details for adapting TLM and TTRL to VLA models.

For TLM, we follow the original formulation and perform test-time adaptation by minimizing the perplexity of the instruction prompt. Concretely, given a task instruction, we optimize the model parameters to reduce the negative log-likelihood of the instruction tokens, without relying on external supervision or environment rewards. We set the loss weighting coefficient to $\lambda = 0.1$ and use a threshold value of 0 for triggering updates. The policy is updated every 8 environment steps. We apply LoRA to update the policy, using a rank of 32 and a learning rate of 1×10^{-4}.

For TTRL, we adapt the consensus-based test-time reinforcement learning framework to the VLA setting. At each decision step, we sample multiple candidate action tokens from the model to construct a pseudo-label via majority voting. We set the voting group size to 8. The reward function is defined as a binary signal: a reward of 1 is assigned if the sampled action token matches the pseudo-label, and 0 otherwise. Policy updates are performed at every environment step to accommodate the step-wise nature of action execution in real-time settings. We employ LoRA to update the policy parameters, with a rank of 32 and a learning rate of 1×10^{-4}.

S6 Additional Real-world Qualitative Results

This section presents additional qualitative results from real-world scenarios, complementing results in §4.3 and further demonstrating the effectiveness of TT-VLA. Fig. S1 presents three real-world rollouts of the "put banana on plate" task using TT-VLA. In the first episode, the robot initially grasps the banana but places it at an incorrect location. It then re-grasps the banana and successfully places it onto the plate. In the second episode, the robot grasps the banana and moves it to a position behind the plate; the policy subsequently corrects its direction and completes the placement. Similarly, in the third episode, the robot initially moves past to the right side of the plate before adjusting its motion to place the banana correctly. These qualitative results demonstrate TT-VLA's ability to recover from execution errors and handle real-world uncertainties without retraining or human intervention.

Algorithm 1: TT-VLA Pipeline

Input: Pretrained VLA policy π_θ, frozen progress estimator $\Phi(o_{0:t}, l)$, language instruction l, observation horizon H, update interval K, clipping threshold ε, learning rate η

Output: Task actions

for each episode **do**
  Load pretrained VLA policy π_θ; progress $p_0 \leftarrow 0$; buffer $\mathcal{B} \leftarrow \emptyset$; Environment resets to initial state s_0
  for each time step $t = 0, 1, 2, \ldots, T$ **do**
    Sample $a_t \sim \pi_\theta(a_t \mid o_{t-1}, l)$, get $\log \pi_{\theta_{\text{old}}}(a_t \mid o_t)$, and execute a_t
    Get new oberservation o_{t+1}
    Compute $p_t \leftarrow \Phi(o_{0:t}, l)$ ▷ Eq. 6
    Compute $r_t \leftarrow p_t - p_{t-1}$ ▷ Eq. 7
    Store $(o_{t+1}, a_t, r_t, \log \pi_{\theta_{\text{old}}}(a_t \mid o_t))$ in $\mathcal{B}$
    if $t \bmod K = 0$ **then**
      for each $(o_i, a_i, r_i, \log \pi_{\theta_{\text{old}}}(a_i \mid o_i)) \in \mathcal{B}$ **do**
        Compute $r_i(\theta) \leftarrow \exp(\log \pi_\theta(a_i \mid o_i) - \log \pi_{\theta_{\text{old}}}(a_i \mid o_i))$
        Compute $L_i \leftarrow \min(r_i(\theta) \cdot r_i, \text{clip}(r_i(\theta), 1-\varepsilon, 1+\varepsilon) \cdot r_i)$ ▷ Eq. 4
        Update policy parameters $\theta \leftarrow \theta + \eta \nabla_\theta \sum_i L_i$
      Clear buffer $\mathcal{B}$
    end if
  end for

S7 Discussions on Using Test-Time GRPO in VLAs

In TT-VLA, we do not adopt Group Relative Policy Optimization (GRPO) (Shao et al., 2024) due to *two practical constraints* in test-time robotic deployment:

1. GRPO relies on sampling multiple candidate trajectories or actions to update the policy, which introduces significant computational overhead and makes it inefficient for real-time test-time adaptation. Such sampling-based procedures are particularly unsuitable under test-time settings, where latency and responsiveness are critical.

2. In real-world robotic scenarios, sampled ac-

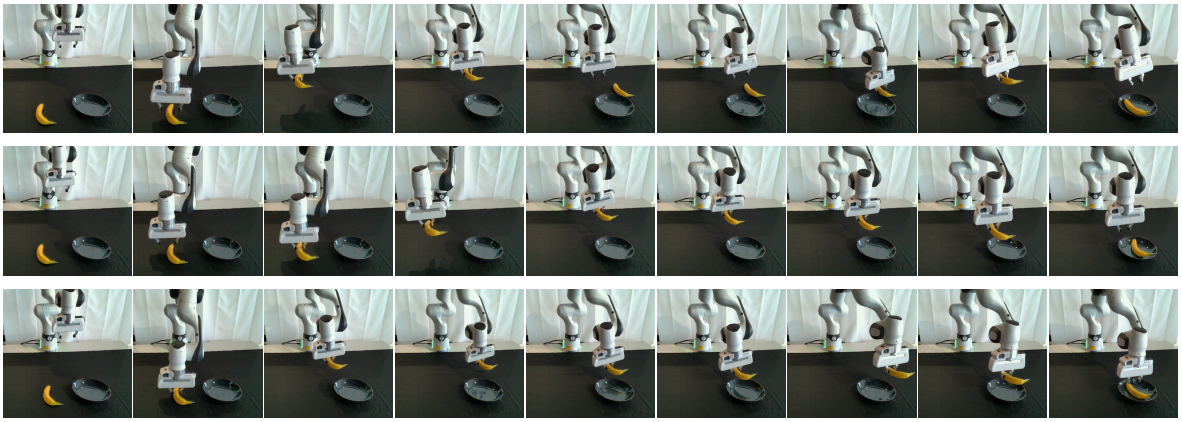


Figure S1: **Additional real-world qualitative results.** Each row shows a real-world episode of the "put banana on plate" task, illustrating how TT-VLA adapts online to execution deviations and successfully completes the task using progress-based reward feedback.

tions inevitably interact with the physical environment (*e.g.*, touching or moving objects). It is thus infeasible to reset the environment to a previous state after each interaction. These constraints make GRPO-style sampling-based optimization impractical for test-time adaptation in physical environments. In fact, that is the practical reason that we redefine the advantage to depend only on the reward obtained from the current action (see §9), as we want to prioritize rapid fitting of the current task rather than state accumulations.

S8 Reproducibility

TT-VLA is implemented in Pytorch (Paszke et al., 2019). Experiments are conducted on NVIDIA RTX 6000 Ada GPUs. To guarantee reproducibility, our full implementation shall be publicly released upon paper acceptance. We provide the pseudo code of TT-VLA in Algorithm 1.

S9 Technical Contributions

Our study presents three principal technical contributions:

- We introduce a test-time reinforcement learning framework for VLA models, enabling robots to adapt their policies on the fly during deployment without requiring retraining or environment resets. This capability directly addresses a key limitation of current VLA systems in real-world robotic settings, where conditions are dynamic and unpredictable.
- To cope with the severe data scarcity and latency constraints at inference time, we propose a dense, progress-based reward that provides stable and task-aligned learning signals at every step, allowing robots to refine their behavior during execution.
- Extensive experiments in both simulated and real-world robotic environments demonstrate that our approach consistently improves the robustness and success rates of existing SFT- and RL-based VLA models, highlighting its practical value for real-world robotic deployment.

S10 Asset License and Consent

All models and datasets used in this work are publicly available. We strictly comply with their original licenses and use them only for non-commercial academic research. The contents of datasets do not represent our views or opinions.

Models. We utilize four open-source models: Nora (MIT license), OpenVLA (MIT license), OpenVLA-RL (MIT license), TraceVLA (MIT license). All licenses permit academic research use; detailed terms are available via the original model repositories.

Datasets. All simulation experiments were conducted in ManiSkill 3. The evaluated tasks are adopted from (Liu et al., 2025a), and detailed task descriptions are provided in §S3. The data (16400 demonstration trajectories) used to warm up the base models is collected following the same procedure as in (Liu et al., 2025a), and is generated automatically.

Consent. Our study does not involve crowdsourcing or human subjects. All results are derived from publicly available models and datasets.

S11 Ethics Concerns

Test-time policy adaptation may increase the risk of unintended or unsafe behaviors, particularly in real-world robotic environments where erroneous actions can result in physical damage, equipment failure, or harm to surrounding objects and people. Because policy updates are performed online and are driven by interaction-derived feedback rather than explicit human supervision, unexpected environmental dynamics or imperfect reward signals may lead to behaviors that deviate from intended task objectives. To mitigate these risks, responsible deployment should incorporate safeguards such as constrained action spaces, explicit safety and termination constraints, and conservative update mechanisms. In addition, human oversight and monitoring remain essential, especially during deployment in safety-critical or unstructured environments, to ensure that adaptive behaviors remain aligned with task goals and safety requirements.

S12 Future Direction

As discussed in §S1.1, owing to the architectural distinctions between discretization-based and diffusion-based approaches, our study primarily focuses on the former. Future work should naturally extend our method to diffusion-based formulations, as TT-VLA provides a generalizable solution. Another promising direction is to utilize test-time adaptation (TTA) methods for effectively augmenting multimodal information.

It should be noted that these discussions on future direction present engineering opportunities rather than insurmountable barriers.

S13 AI Disclosure

We acknowledge the use of GPT-5 for grammar checking only. The model was employed to correct grammatical errors while ensuring the original meaning and intent of the text remained unchanged.