REALIGN-VLA: DEVIATION-GUIDED RECOVERY FOR OFF-THE-SHELF VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action models have shown strong potential for language-conditioned robotic manipulation, but reliable deployment requires the ability to recover from execution failures, not merely to generate actions. In practical manipulation, routine visual uncertainty, contact errors, and action drift can turn small deviations into task failure, while recovery strategies based on VLA fine-tuning or auxiliary models limit scalability in open-ended and changing environments. We propose ReAlign-VLA, a model-update-free recovery framework that couples internal-state runtime monitoring with deviation-conditioned intervention. Rather than training a separate failure detector or invoking an external VLM planner, ReAlign-VLA constructs a progress-conditioned profile of successful executions from the frozen VLA's hidden states. During deployment, it extracts hidden states through a lightweight forward hook and computes a sliding-window normalized deviation score to identify when the current execution departs from successful behavior. The deviation score then gates a tiered recovery mechanism, escalating from action-level correction for physical execution errors to deterministic sub-goal refinement for semantic or planning ambiguity. Progress-aware locking and early abort make the intervention conservative, preserving near-successful trajectories while discarding attempts that remain far from the success profile. Experiments on four LIBERO benchmark suites show that ReAlign-VLA consistently improves an unmodified OpenVLA baseline in both task success and execution efficiency.

1 INTRODUCTION

Vision-Language-Action (VLA) models such as RT-2 Zitkovich et al. (2023), Octo Team et al. (2024), and OpenVLA Kim et al. (2024) map visual observations and natural-language instructions directly to continuous robot actions. On standardized benchmarks such as LIBERO Liu et al. (2023), fine-tuned VLAs achieve strong average-case performance across diverse manipulation tasks. However, reliable deployment requires more than producing plausible actions. In practical execution, small visual uncertainties, contact errors, and accumulated action deviations can cause the robot to leave a successful trajectory, while the VLA often continues acting without recognizing that recovery is needed Gu et al. (2026). Across the four LIBERO suites, a fine-tuned OpenVLA-7B Kim et al. (2024) still fails in 21–53% of tasks. These failures commonly appear as physical imprecision, visual misidentification, and long-horizon trajectory drift, suggesting that runtime failure awareness and recovery are essential for deploying VLAs beyond average-case benchmark success.

Existing recovery strategies can be broadly categorized into training-based recovery and external-supervisor-based recovery. Training-based methods (Fig. 1(a)) adapt the VLA or an additional recovery module using failure-correction data, reinforcement learning, or synthetic failure demonstrations. Representative methods such as TCoT Li et al. (2026), PLD Xiao et al. (2025), and FailSafe Lin et al. (2025) improve recovery by incorporating failure-handling capability into the model or its companion module. However, such recovery is tied to additional data collection and model updates, making it less scalable when tasks, objects, or visual conditions change. External-supervisor-based methods avoid directly updating the VLA by adding an auxiliary reasoning module at inference time, as shown in Fig. 1(b). Methods such as FPC-VLA Yang et al. (2026b) and

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

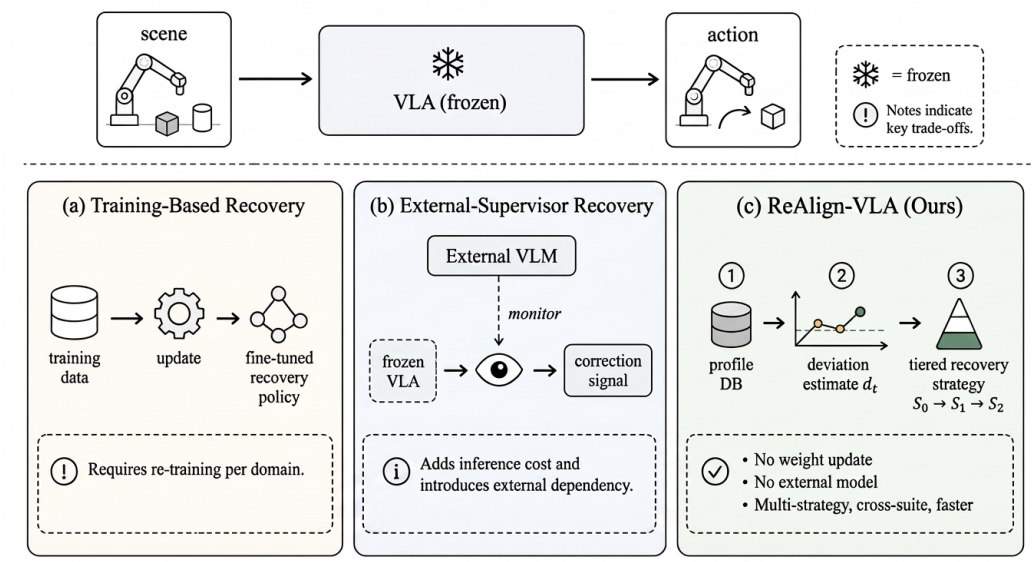

Figure 1: Comparison of VLA failure recovery paradigms. (a) Training-based recovery modifies the VLA through additional training. (b) External-supervisor recovery adds an auxiliary model alongside the frozen VLA. (c) ReAlign-VLA (Ours) reads internal deviation signals from the frozen VLA, requiring neither weight updates nor external models.

AHA Duan et al. (2024) use a VLM or supervisor to predict failures, explain failure causes, or produce corrective actions. Although these methods preserve the original VLA weights, they introduce additional components into the control loop and make recovery dependent on the cost, latency, and reliability of the auxiliary model.

Despite their different implementations, both paradigms obtain recovery capability from outside the frozen VLA, either through additional training or through an auxiliary inference-time supervisor. Consequently, the VLA's internal execution state remains largely underused as a source of failure evidence. Yet successful executions often follow regular, progress-dependent patterns in representation space, with hidden states reflecting object grounding, action intent, and task progress. Prior studies on VLA representation probing Lu et al. (2025) and internal-state-based failure detection Gu et al. (2026) support this observation. When execution deviates, these patterns may be disrupted before the failure becomes externally apparent. This suggests formulating recovery as monitoring deviations from a progress-conditioned success profile and triggering intervention when the deviation becomes significant.

In this paper, we propose ReAlign-VLA, a model-update-free recovery framework that couples internal-state runtime monitoring with deviation-conditioned intervention, as illustrated in Fig. 1(C). Rather than training a separate failure detector or invoking an external VLM planner, ReAlign-VLA uses the frozen VLA's own hidden states as the basis for failure awareness. It first collects hidden states from successful executions and organizes them into a progress-conditioned success profile, which provides a stage-aware reference of how the VLA behaves internally when execution is on track. During deployment, ReAlign-VLA extracts hidden states through a lightweight forward hook and compares the current execution against the success profile. To avoid reacting to transient noise, it computes a sliding-window normalized deviation score that captures persistent departures from successful behavior. This score serves as the runtime signal for deciding when intervention is needed and which recovery level should be used. For low-level physical execution errors, ReAlign-VLA applies action-level correction through re-execution and action-space adjustment. For semantic or planning ambiguity, it escalates to deterministic sub-goal refinement, which decomposes the original instruction into simpler manipulation sub-goals without relying on an external language model. Progress-aware locking and early abort further make the intervention conservative by preserving near-successful trajectories and discarding attempts that remain far from the success profile.

Our contributions are summarized as follows:

1. We introduce ReAlign-VLA, a model-update-free recovery framework for off-the-shelf VLA models. It couples internal-state runtime monitoring with deviation-conditioned intervention, enabling failure recovery without updating VLA weights, training an additional failure detector, or invoking an external VLM planner.

2. We propose progress-conditioned success profiling for VLA execution. By collecting hidden states from successful rollouts, ReAlign-VLA builds a stage-aware reference of the frozen VLA's internal behavior, allowing execution failures to be detected as deviations from successful task progress.

3. We develop a deviation-gated tiered recovery mechanism with progress-aware safeguards. The recovery process escalates from action-level correction for physical execution errors to deterministic sub-goal refinement for semantic or planning ambiguity, while progress-aware locking and early abort preserve near-successful trajectories and discard unpromising attempts.

2 Related Work

2.1 Vision-Language-Action Models

The integration of large language models with robot control, pioneered by works such as SayCan Ahn et al. (2022) and extended by Vision-Language-Action (VLA) models that directly map perceptual inputs to continuous action outputs Hu et al. (2023). OpenVLA Kim et al. (2024) is a 7B-parameter open-source VLA built on a Prismatic VLM backbone Karamcheti et al. (2024) and fine-tuned on the Open-X Embodiment dataset Collaboration et al. (2023), achieving strong performance across diverse manipulation benchmarks. RT-2 Zitkovich et al. (2023) and RT-1 Brohan et al. (2022) demonstrate that vision-language models co-fine-tuned on robot trajectory data can transfer web-scale visual knowledge to physical tasks. Octo Team et al. (2024) introduces a transformer-based generalist policy trained on multiple robot embodiments. Recent models such as FAST Pertsch et al. (2025), GR00T Bjorck et al. (2025), and OpenVLA-OFT Kim et al. (2025), and ChatVLA Zhou et al. (2025b) further advance the field. Despite their impressive average-case performance, these models share a critical limitation: they lack any built-in mechanism for detecting when they have failed or for recovering from errors. As a result, when a VLA misidentifies an object, misses a grasp, or drifts off a long-horizon trajectory, it continues executing without awareness of the failure, a property that limits their reliability in real-world deployment.

2.2 Failure Detection and Recovery

Efforts to make VLAs more reliable span detection and recovery, and can be organized by how they relate to the VLA's own parameters. Methods that train additional components include SAFE Gu et al. (2026), which learns multitask probes on intermediate representations to classify failures, and Hide-and-Seek Park et al. (2026), which uses contrastive learning to discover failure signals from trajectory-level labels. On the recovery side, TCoT Li et al. (2026) trains a hierarchical planning module with global-local switching; PLD Xiao et al. (2025) employs residual RL to distill recovery policies; and FailSafe Lin et al. (2025) fine-tunes a VLM on synthetic failure data. These approaches improve robustness but require task-specific training data and model updates.

Methods that avoid modifying the VLA instead invoke external reasoning at inference time. AHA Duan et al. (2024) and INSIGHT Karli et al. (2025) use VLMs or uncertainty signals for failure detection. FPC-VLA Yang et al. (2026b) deploys a VLM supervisor to predict failures and guide action fusion. ProbeAct Zhang et al. (2026) achieves runtime recovery via a hidden-state probe that predicts 3D positions for control barrier function correction, though the probe itself requires supervised training. AFI Xu et al. (2025) uses 3D affordance fields to escape memory traps. RACER Dai et al. (2025) and CoT-VLA Zhao et al. (2025) explore language-guided recovery and chain-of-thought reasoning, respectively. These methods preserve VLA weights but add inference overhead and an external dependency to the control loop.

Tri-Info Yang et al. (2026a) and Diffusion Policy Chi et al. (2025) approach related problems from complementary angles: information-theoretic failure prediction and diffusion-based action generation.

Our method differs from both categories. Rather than training a detector or invoking an external reasoning module, ReAlign-VLA monitors the drift between live hidden states and an offline progress-conditioned profile, triggering tiered recovery only when deviation indicates that the current execution has left the success distribution.

2.3 Internal State Probing and Representation Analysis

Recent work shows that VLA hidden states encode structured, task-relevant information. Lu et al. Lu et al. (2025) probe OpenVLA layers to uncover representations of object properties, spatial relations, and action states. Karamcheti et al. Karamcheti et al. (2024) establish the VLM design foundation behind OpenVLA. These studies demonstrate that VLA internal representations are not opaque, and can be exploited for monitoring without training. Larger-scale efforts such as π_0 Black et al. (2024), $\pi_{0.5}$ Intelligence et al. (2025), and BridgeData V2 Walke et al. (2023) focus on policy capability rather than runtime reliability. Our method is the first to be simultaneously model-update-free, external-model-free, and equipped with tiered multi-strategy intervention driven by a unified deviation signal transferred zero-shot across suites.

3 Method

3.1 Problem Formulation

We consider a frozen Vision-Language-Action (VLA) model π_θ that maps a visual observation $\mathbf{o}_t$ and a natural-language instruction ℓ to a continuous robot action $\mathbf{a}_t \in \mathbb{R}^7$ at timestep t. The action parameterizes end-effector translation, rotation, and gripper control. An episode is specified by an instruction ℓ and an initial environment state, and terminates when the task is completed or when the maximum horizon $T_{\max}$ is reached. Let $\tau = \{(\mathbf{o}_t, \mathbf{a}_t)\}_{t=1}^{T}$ denote the resulting trajectory. The episode is successful if the final state satisfies the goal condition specified by ℓ, and is considered failed otherwise.

This work studies execution-time failure recovery for off-the-shelf VLA models. At each timestep, the recovery module observes the current execution history and has access to the VLA's internal hidden state $\mathbf{h}_t$, which is produced during the standard forward pass of π_θ. The goal is to decide whether the current execution is deviating from successful behavior and, when necessary, to replace the original action $\mathbf{a}_t$ with a corrected action $\hat{\mathbf{a}}_t$ or a refined instruction-conditioned action so that the recovered trajectory is more likely to satisfy the task goal.

We impose three constraints that distinguish this setting from training-based or externally supervised recovery. First, the VLA parameters θ remain fixed throughout deployment. Second, no additional learned failure detector, planner, reward model, or external VLM is invoked at inference time. Third, the only offline information available to the recovery module is a set of successful baseline trajectories $\mathcal{D}_s = \{\tau_i \mid \text{success}(\tau_i) = 1\}$, from which lightweight statistical references may be constructed. Under these constraints, the recovery problem is to use the VLA's existing forward-pass representations to detect execution deviation and guide intervention without modifying the VLA itself.

Formally, a recovery policy $\mathcal{R}$ takes the execution history, the instruction, the VLA-predicted action, and the internal representation as input,

$$\hat{\mathbf{a}}_t = \mathcal{R}(\mathbf{o}_{1:t}, \mathbf{a}_{1:t}, \ell, \mathbf{h}_{1:t}; \mathcal{D}_s), \tag{1}$$

where $\hat{\mathbf{a}}_t = \mathbf{a}_t$ when no intervention is needed. The objective is to improve the probability of task success over the unmodified VLA while keeping intervention conservative, so that recovery is activated only when the execution shows evidence of deviation from successful behavior.

3.2 Overview of ReAlign-VLA

As shown in Figure 2, ReAlign-VLA consists of three core components. First, progress-conditioned success profiling constructs a stage-aware reference of successful VLA execution from hidden states

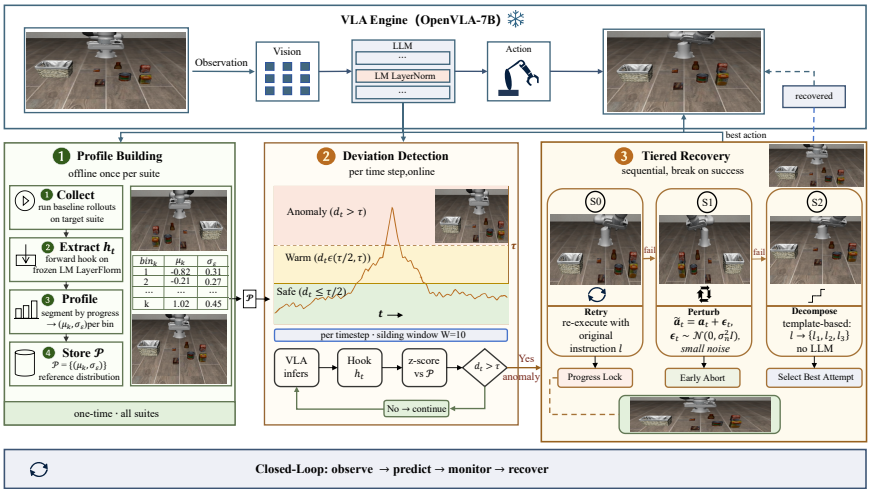

Figure 2: Overview of ReAlign-VLA. Phase 1 (Profile Building) runs offline to construct a statistical baseline from successful executions. Phase 2 (Deviation Detection) runs online, monitoring hidden-state drift and triggering recovery when deviation exceeds a learned threshold. Phase 3 (Tiered Recovery) escalates from action-level correction to deterministic sub-goal refinement under progress-aware safeguards.

collected during successful rollouts. Second, sliding-window deviation scoring monitors whether the current execution departs from this reference during deployment. Third, deviation-gated tiered recovery intervenes when persistent deviation is detected, escalating from action-level correction to deterministic sub-goal refinement under progress-aware safeguards.

This design differs from recovery methods that learn an additional failure classifier or invoke an external VLM planner. ReAlign-VLA uses only the internal states produced by the frozen VLA during its standard forward pass. The success profile is built offline from successful trajectories, while all online decisions are made by comparing the current internal state against this profile. In this way, recovery is driven by the VLA's own execution dynamics rather than by an additional learned component.

3.3 Progress-Conditioned Success Profiling

ReAlign-VLA first constructs a statistical reference of successful execution using the frozen VLA's internal representations. Given a VLA model π_θ, we execute it without intervention and collect successful baseline rollouts. At each timestep t, we register a forward hook on the final layer normalization of the VLA's language-model backbone to extract the hidden state $\mathbf{h}_t \in \mathbb{R}^{4096}$. This hook does not introduce additional model computation, since the layer normalization is already evaluated during the standard forward pass.

Directly comparing raw hidden states is noisy and inefficient. We therefore map each $\mathbf{h}_t$ into a compact feature vector $\mathbf{f}_t$ that captures both representation-level and action-level execution information. Specifically, we partition $\mathbf{h}_t$ into equally sized segments and compute first- and second-order statistics for each segment, yielding distributional features of the hidden representation. We further append physical features derived from the predicted action $\mathbf{a}_t$, such as end-effector motion, gripper state, and temporal action changes, together with global statistics of $\mathbf{h}_t$. This produces a compact state-action feature that is more stable for deviation estimation than the original high-dimensional hidden vector.

To make the reference profile aware of task progress, each successful trajectory is divided into K bins according to normalized execution progress. For each progress bin k and feature dimension j, we compute the empirical mean $\mu_{k,j}$ and standard deviation $\sigma_{k,j}$ across successful rollouts. The resulting profile is defined as

$$\mathcal{P} = \{(\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k)\}_{k=1}^{K}. \tag{2}$$

The profile $\mathcal{P}$ provides a progress-aware statistical reference of how the VLA behaves internally during successful execution. It serves as the reference distribution for the online deviation score introduced next.

3.4 Sliding-Window Deviation Scoring

During deployment, ReAlign-VLA uses the success profile as a reference for runtime monitoring. At each timestep t, the same forward hook extracts the hidden state $\mathbf{h}_t$, which is converted into the compact feature $\mathbf{f}_t$. The current progress bin is estimated from normalized execution progress, and the corresponding profile statistics $(\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k)$ are retrieved from $\mathcal{P}$.

The deviation score d_t is defined as the sliding-window average of the normalized distance between the observed feature vector and the success profile,

$$d_t = \frac{1}{W} \sum_{i=t-W+1}^{t} \left\| \frac{\mathbf{f}_i - \boldsymbol{\mu}_{k(i)}}{\boldsymbol{\sigma}_{k(i)} + \epsilon} \right\|_2, \tag{3}$$

where W is the window size, $k(i)$ maps timestep i to its progress bin, and ϵ is a small constant for numerical stability.

The sliding window suppresses transient fluctuations caused by stochastic action sampling, momentary occlusion, or visual noise. The normalization by $\boldsymbol{\sigma}_{k(i)}$ accounts for the natural variability of each feature dimension, preventing stable dimensions from dominating the score and noisy dimensions from saturating it. Low values of d_t indicate that the current internal state matches the profile of successful execution, while high values indicate that the execution is drifting away from the success profile. When $d_t \leq \tau$, ReAlign-VLA executes the original VLA action without intervention. When $d_t > \tau$, the deviation is considered persistent enough to activate the recovery module.

3.5 Deviation-Gated Tiered Recovery

The deviation score is not used as a passive alarm. It gates a tiered recovery policy that matches the intervention strength to the type and persistence of execution deviation. The policy starts from conservative action-level correction and escalates to semantic-level instruction refinement only when lower-level correction is insufficient.

Action re-execution. The first recovery level re-executes the task from the current environment state using the original instruction. This intervention is designed for low-level physical execution errors where the high-level plan remains valid but the realized action is perturbed, such as minor gripper misalignment, slip, or transient collision-induced offsets. Since it does not alter the instruction or the VLA parameters, it is the least invasive recovery strategy.

Action-space adjustment. When re-execution does not reduce the deviation, the VLA may be trapped in a narrow region of its action distribution and repeatedly produce similar failing actions. ReAlign-VLA then applies action-space adjustment by adding small isotropic Gaussian perturbations to the predicted action,

$$\tilde{\mathbf{a}}_t = \mathbf{a}_t + \boldsymbol{\epsilon}_t, \qquad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \sigma_n^2 \mathbf{I}). \tag{4}$$

This adjustment nudges the trajectory toward a neighboring action region while preserving the original policy structure. It is intended to recover from local action errors without changing the semantic interpretation of the instruction.

Deterministic sub-goal refinement. If action-level correction remains insufficient, ReAlign-VLA escalates to semantic-level intervention. The original high-level instruction is decomposed into a sequence of fine-grained manipulation sub-goals using predefined task-agnostic templates. This module invokes no external LLM, planner, or learned component. For pick-and-place tasks, the template decomposes the instruction into a sequence of manipulation primitives, including *approach*, *grasp*, *lift*, *move*, *place*, and *release*. Object names and target locations are parsed from the original instruction through deterministic keyword matching. For non-pick-and-place tasks, such as opening a

drawer or turning a knob, we use corresponding manipulation templates following the same deterministic design. The decomposed instruction changes only the language input to the VLA, while the visual observation, action head, and model weights remain unchanged. This level is designed for semantic ambiguity or visual confusion, where repeating the same instruction is unlikely to change the grounding behavior.

3.6 Progress-Aware Safeguards

Tiered recovery can improve robustness only if unnecessary interventions are avoided. ReAlign-VLA therefore uses two progress-aware safeguards to regulate when recovery should continue, switch, or stop.

Progress-aware locking. When the deviation score is low and the end-effector is making measurable progress toward the task-relevant object or target region, ReAlign-VLA locks the current recovery strategy. This prevents transient sensor noise or short-lived deviation spikes from causing unnecessary strategy switches during an otherwise successful trajectory. Formally, the current strategy is locked when

$$d_t < d_{\text{lock}} \quad \text{and} \quad \Delta p_t > 0, \tag{5}$$

where Δp_t denotes recent task-progress improvement estimated from end-effector motion or distance reduction to the target.

Early abort. When the deviation score is critically high and task progress remains near zero, the current attempt is unlikely to recover. ReAlign-VLA then terminates the attempt and escalates to the next recovery level. Formally, early abort is activated when

$$d_t > d_{\text{abort}} \quad \text{and} \quad \Delta p_t \leq \eta, \tag{6}$$

where d_{abort} is the abort threshold and η is a small progress tolerance. This mechanism avoids spending the full episode horizon on trajectories that remain far from the success profile and show little evidence of task progress.

Together, progress-aware locking and early abort make intervention conservative. The former preserves near-successful trajectories, while the latter prevents the system from repeatedly executing unpromising attempts.

The complete procedure alternates standard frozen-VLA execution with the deviation test in Eq. 3; only persistent deviations invoke tiered recovery, after which execution resumes from the new observation.

4 Experiments

4.1 Experimental Setup

Benchmark and base model. We evaluate ReAlign-VLA on all four LIBERO suites Liu et al. (2023); Zhou et al. (2025a), including Spatial, Object, Goal, and LIBERO-10. Spatial focuses on spatial reasoning and precise placement, Object emphasizes visually similar object discrimination, Goal covers diverse goal configurations, and LIBERO-10 contains long-horizon manipulation tasks. Each suite contains 10 tasks, and we evaluate 200 instruction-scene pairs per suite following the standard protocol. We use OpenVLA-7B Kim et al. (2024) as the base model, fine-tuned separately on each suite. Spatial, Object, and Goal are evaluated with three random seeds, while LIBERO-10 is evaluated with one seed due to its substantially longer episode duration.

Metrics. We report success rate (SR) as the primary metric. To measure execution efficiency, we also report step overhead, defined as $\text{steps}_{\text{ours}}/\text{steps}_{\text{baseline}}$. An overhead below $1.0\times$ indicates that ReAlign-VLA completes evaluation using fewer execution steps than the unmodified baseline.

Hyperparameters. All hyperparameters are calibrated by offline grid search on the Spatial suite and then used without modification on the other three suites. The selected values are sliding window size $W = 10$, deviation threshold $\tau = 0.8$, warmup steps $= 25$, early abort threshold $d_{\text{abort}} = 1.8$, progress lock threshold $d_{\text{lock}} = 0.6$, action noise scale $\sigma_n = 0.03$, and $K = 10$ progress bins.

Suite	Seeds	Base	Ours	Δ	Overhead
Spatial	3	78.8%	89.5%	+10.7%	0.90×
Object	3	65.5%	86.5%	+21.0%	0.89×
Goal	3	74.2%	92.2%	+18.0%	0.84×
LIBERO-10	1	47.5%	72.5%	+25.0%	0.87×

Table 1: Matched comparison with the unmodified OpenVLA-7B baseline on the four LIBERO suites. ReAlign-VLA improves success rate while keeping step overhead below $1.0\times$.

Method	Spatial	Object	Goal	LIBERO-10
Ours baseline	78.8	65.5	74.2	47.5
CoT-VLA Zhao et al. (2025)	87.5	**91.6**	87.6	69.0
ThinkAct Huang et al. (2026)	88.3	91.4	87.1	70.9
ReAlign-VLA	**89.5**	86.5	**92.2**	**72.5**

Table 2: Comparison to published LIBERO results. Published numbers may use different evaluation protocols and rollout sets, and are provided as contextual references.

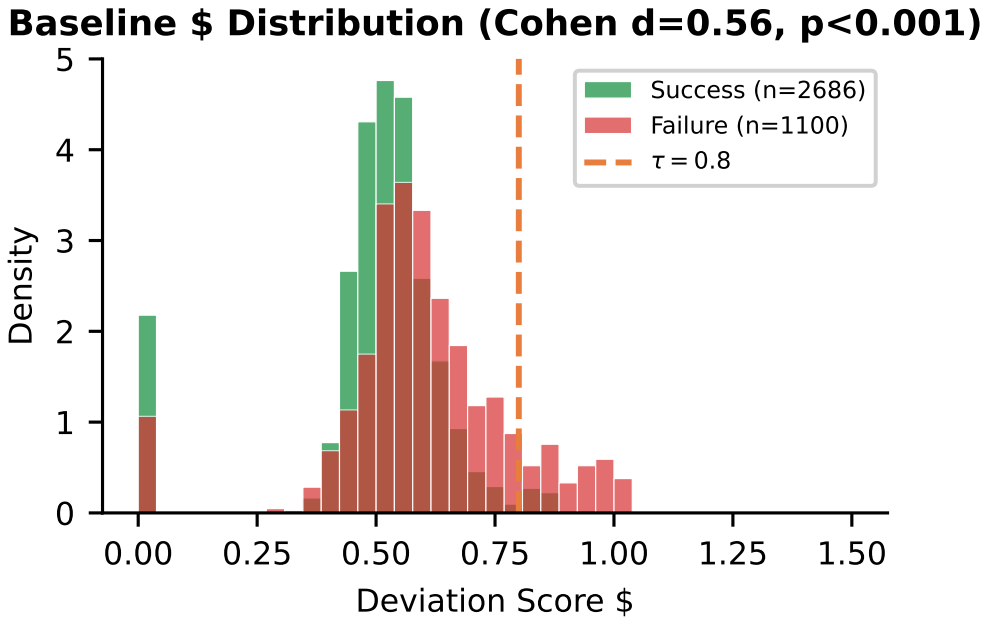

Figure 3: Deviation score distributions for successful and failed baseline trajectories. The two distributions are significantly separated at $\tau = 0.8$.

The grid search is conducted on held-out validation trajectories from the baseline rollouts collected during success profiling, without online tuning on the evaluation tasks. Additional implementation details, hyperparameter sensitivity analysis, and deterministic instruction templates are provided in the supplementary material.

4.2 Main Results and Comparison

Table 1 reports the matched comparison between ReAlign-VLA and the unmodified OpenVLA-7B baseline under the same evaluation protocol. ReAlign-VLA consistently improves success rate across all four LIBERO suites, with gains ranging from $+10.7\%$ on Spatial to $+25.0\%$ on LIBERO-10. The step overhead remains below $1.0\times$ across all suites, indicating that the improved task success does not come at the cost of longer execution.

Ablation	Spatial	Object	Goal	Long
Full method	$+10.7$	$+21.0$	$+18.0$	$+25.0$
Random retry	$+16.0$	$+0.0$	$+18.5$	$+14.0$
Single strategy	–	-3.0	–	–
No progress lock	$+13.0$	–	–	$+22.5$
No early abort	–	–	$+16.5$	$+28.0$

Table 3: Ablation results measured by ΔSR relative to the baseline. Different components contribute differently across failure modes.

The largest improvement appears on LIBERO-10, where the baseline success rate is lowest. This suggests that deviation-guided recovery is particularly useful for long-horizon manipulation, where small execution errors can accumulate over many steps. The strong gains on Object and Goal further indicate that ReAlign-VLA is not limited to a single failure type, but can handle both visual confusion and goal-conditioned execution errors. Per-seed results are provided in the supplementary material and show stable improvements across the three multi-seed suites.

Table 2 further compares ReAlign-VLA with representative published LIBERO results. Since published results may differ in evaluation protocol, rollout set, and implementation details, this table is intended as a contextual comparison rather than a strictly controlled head-to-head benchmark. The matched OpenVLA baseline in Table 1 therefore serves as the primary controlled comparison.

4.3 Deviation Signal Analysis

On Spatial, failed baseline executions have consistently higher deviation scores than successful executions (Figure 3): the respective means are 0.600 and 0.495, with Cohen's $d = 0.56$ and $p < 0.001$. At the per-step level, the detector is intentionally conservative, with a false positive rate of 2.1% and a true positive rate of 12.5% at $\tau = 0.8$. Since recovery is triggered by persistent sliding-window deviation rather than isolated frame-level spikes, this design prioritizes avoiding unnecessary interventions on trajectories that would otherwise succeed. Hyperparameter sensitivity results are provided in the supplementary material.

4.4 Ablation Study

We conduct ablations to isolate the effects of deviation-guided recovery, tiered intervention, and progress-aware safeguards. Table 3 reports the success-rate change relative to the baseline. These results show that different failure modes require different recovery levels. Random retry improves Spatial, Goal, and LIBERO-10, but gives no gain on Object, indicating that visual confusion cannot be resolved by re-execution alone. In contrast, using a single strategy on Object reduces performance below the baseline, suggesting that deviation detection must be paired with sufficiently diverse recovery actions.

The safeguards also exhibit task-dependent effects. Removing progress lock reduces the gain on LIBERO-10, where long-horizon executions are more sensitive to unnecessary strategy switches. Removing early abort slightly improves LIBERO-10 but reduces Goal, indicating that a single abort threshold may be suboptimal across horizons. The supplementary material includes the full ablation matrix with step overhead and a heatmap visualization.

5 Conclusion

We presented ReAlign-VLA, a model-update-free framework combining progress-conditioned success profiling, runtime deviation monitoring, and tiered recovery. Across four LIBERO suites, it consistently improves the OpenVLA-7B baseline while keeping step overhead below $1.0\times$. The results show that frozen-VLA internal states can support reliable recovery without additional model training or external reasoning modules.

AI USE STATEMENT

Generative AI tools were used to assist with language polishing and formatting the manuscript for ICLR. The authors reviewed all AI-assisted edits and take responsibility for the final content of the paper, including its text, claims, and artifacts.

REPRODUCIBILITY STATEMENT

The method definition, deviation score, recovery procedure, implementation details, hyperparameters, evaluation protocol, and ablation settings needed to reproduce the reported experiments are provided in the Method and Experiments sections. The paper also reports per-suite and per-task results to make the evaluation and comparisons auditable.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

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 preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

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

Cheng Chi, Siyuan Feng, Zhenjia Xu, Eric A Cousineau, Benjamin Burchfiel, Shuran Song, et al. Visuomotor policy learning via action diffusion, September 4 2025. US Patent App. 18/594,842.

OX-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 1(2), 2023.

Yinpei Dai, Jayjun Lee, Nima Fazeli, and Joyce Chai. Racer: Rich language-guided failure recovery policies for imitation learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15657–15664. IEEE, 2025.

Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. Aha: A vision-language-model for detecting and reasoning over failures in robotic manipulation. *arXiv preprint arXiv:2410.00371*, 2024.

Qiao Gu, Yuanliang Ju, Shengxiang Sun, Igor Gilitschenski, Haruki Nishimura, Masha Itkina, and Florian Shkurti. Safe: Multitask failure detection for vision-language-action models. *Advances in Neural Information Processing Systems*, 38:40041–40076, 2026.

Yafei Hu, Quanting Xie, Vidhi Jain, Jonathan Francis, Jay Patrikar, Nikhil Keetha, Seungchan Kim, Yaqi Xie, Tianyi Zhang, Hao-Shu Fang, et al. Toward general-purpose robots via foundation models: A survey and meta-analysis. *arXiv preprint arXiv:2312.08782*, 2023.

Chi-Pin Huang, Yueh-Hua Wu, Min-Hung Chen, Frank Wang, and Fu-En Yang. Thinkact: Vision-language-action reasoning via reinforced visual latent planning. *Advances in Neural Information Processing Systems*, 38:82782–82802, 2026.

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 preprint arXiv:2504.16054*, 2025.

Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models. In *Forty-first International Conference on Machine Learning*, 2024.

Ulas Berk Karli, Ziyao Shangguan, and Tesca FItzgerald. Insight: Inference-time sequence introspection for generating help triggers in vision-language-action models. *arXiv preprint arXiv:2510.01389*, 2025.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

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

Xiang Li, Ya-Li Li, Yuan Wang, Huaqiang Wang, and Shengjin Wang. Tcot: Trajectory chain-of-thoughts for robotic manipulation with failure recovery in vision-language-action model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 6486–6494, 2026.

Zijun Lin, Jiafei Duan, Haoquan Fang, Dieter Fox, Ranjay Krishna, Cheston Tan, and Bihan Wen. Failsafe: Reasoning and recovery from failures in vision-language-action models. *arXiv preprint arXiv:2510.01642*, 2025.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Hong Lu, Hengxu Li, Prithviraj Singh Shahani, Stephanie Herbers, and Matthias Scheutz. Probing a vision-language-action model for symbolic states and integration into a cognitive architecture. In *2025 IEEE International Conference on AI and Data Analytics (ICAD)*, pp. 1–8. IEEE, 2025.

Seongheon Park, Wendi Li, Changdae Oh, Samuel Yeh, Zsolt Kira, Michael Hagenow, and Sharon Li. Hide-and-seek in trajectories: Discovering failure signals for vla runtime monitoring. *arXiv preprint arXiv:2605.30834*, 2026.

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

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pp. 1723–1736. PMLR, 2023.

Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Yuqi Xie, Fengyuan Hu, Jimmy Wu, Zhengyi Luo, Linxi Fan, et al. Self-improving vision-language-action models with data generation via residual rl. *arXiv preprint arXiv:2511.00091*, 2025.

Siyu Xu, Zijian Wang, Yunke Wang, Chenghao Xia, Tao Huang, and Chang Xu. Affordance field intervention: Enabling vlas to escape memory traps in robotic manipulation. *arXiv preprint arXiv:2512.07472*, 2025.

Jinghan Yang, Yunchao Zhang, Wang Yuan, Haolun Wan, Jiaming Zhang, Zhengyang Hu, and Yanchao Yang. Tri-info: Generalizable, interpretable failure prediction for vla models via information theory. *arXiv preprint arXiv:2606.19998*, 2026a.

Yifan Yang, Zhixiang Duan, Tianshi Xie, Fuyu Cao, Pinxi Shen, Peili Song, Chenyang Zhao, Piaopiao Jin, Guokang Sun, Shaoqing Xu, et al. Fpc-vla: A vision-language-action framework with a supervisor for failure prediction and correction. *Expert Systems with Applications*, pp. 131742, 2026b.

Fan Zhang, Seongbin Park, Baharan Mirzasoleiman, Shariar Talebi, and Nader Sehatbakhsh. Probe-act: Probe-guided training-free failure recovery in vision-language-action models. *arXiv preprint arXiv:2606.09740*, 2026.

Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. *arXiv preprint arXiv:2503.22020*, 2025.

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. Libero-pro: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025a.

Zhongyi Zhou, Yichen Zhu, Minjie Zhu, Junjie Wen, Ning Liu, Zhiyuan Xu, Weibin Meng, Yaxin Peng, Chaomin Shen, Feifei Feng, et al. Chatvla: Unified multimodal understanding and robot control with vision-language-action model. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5377–5395, 2025b.

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 *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.