LA4VLA: LEARNING TO ACT WITHOUT SEEING VIA LANGUAGE-ACTION PRETRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) models are commonly pretrained on robot demonstrations by jointly mapping visual observations and language instructions to actions. However, dense visual-action supervision can dominate the comparatively sparse language-action signal. As a result, policies may rely on visual shortcuts rather than learn how language conditions action execution, making them sensitive to visual variations. To address this limitation, we propose LA4VLA, a language-action pretraining framework that enables policies to acquire language-conditioned action priors without visual observations. These priors capture reusable manipulation skills shared across tasks and scenes, reducing reliance on scene-specific visual cues. Specifically, LA4VLA decomposes expert demonstration trajectories into atomic action segments and pairs each segment with a corresponding low-level action description. This yields LA-33K, a dataset of 33K Language-Action (LA) episodes derived entirely from existing demonstrations without additional robot data collection. We further develop LA4VLA-1B, a lightweight 1B-parameter VLA model, and investigate three paradigms for incorporating language-action supervision into VLA learning: LA-only pretraining, sequential LA-to-VLA pretraining, and mixed LA-VLA pretraining. Across simulation and real-world tasks, LA-pretrained policies consistently outperform matched VLA-pretrained counterparts, while combining LA and VLA supervision leads to further gains. In particular, mixed LA-VLA pretraining improves the average success rate of LA4VLA-1B over the no-pretraining baseline by up to 17.8 and 45.0 percentage points in simulation and real-world tasks, respectively. These results establish LA4VLA as an effective and complementary pretraining strategy for building stronger and more robust VLA policies. We release our code, data, and model weights to facilitate future research on language-action pretraining for VLA models.

1 INTRODUCTION

Vision-Language-Action (VLA) models (Kim et al., 2024; Black et al., 2024; Bjorck et al., 2025; Wang et al., 2026b; Lin et al., 2025; 2026a; Wang et al., 2026c;a; Zuo et al., 2026) have recently emerged as a promising paradigm for general-purpose robot manipulation. These models are typically trained on large-scale robot demonstrations (Khazatsky et al., 2024; O'Neill et al., 2024; Walke et al., 2023; Fang et al., 2026a), where each trajectory contains visual observations, action sequences, and a high-level language instruction specifying the task goal. By learning to predict actions from the current visual observations and the given instruction, this multimodal joint training paradigm enables robots to perform diverse manipulation tasks from language commands.

Despite its effectiveness, the joint VLA training paradigm introduces an inherent asymmetry in how vision and language contribute to action learning. This asymmetry appears at two levels. First, **at the data level**, a single robot trajectory can contain hundreds or even thousands of image-action pairs, but is typically paired with only one high-level task instruction. The instruction often specifies only the overall task goal, while visual observations and robot states vary continuously throughout the trajectory. As a result, vision-action supervision is temporally dense and dynamically changing, whereas language-action supervision has much less semantic variation and lacks explicit alignment to local action stages. Second, **at the model-input level**, visual observations are usually encoded into hundreds of image tokens, often outnumbering the tokens of language instructions by one to two orders of magnitude. Together, these data-level and input-level asymmetries can bias action

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

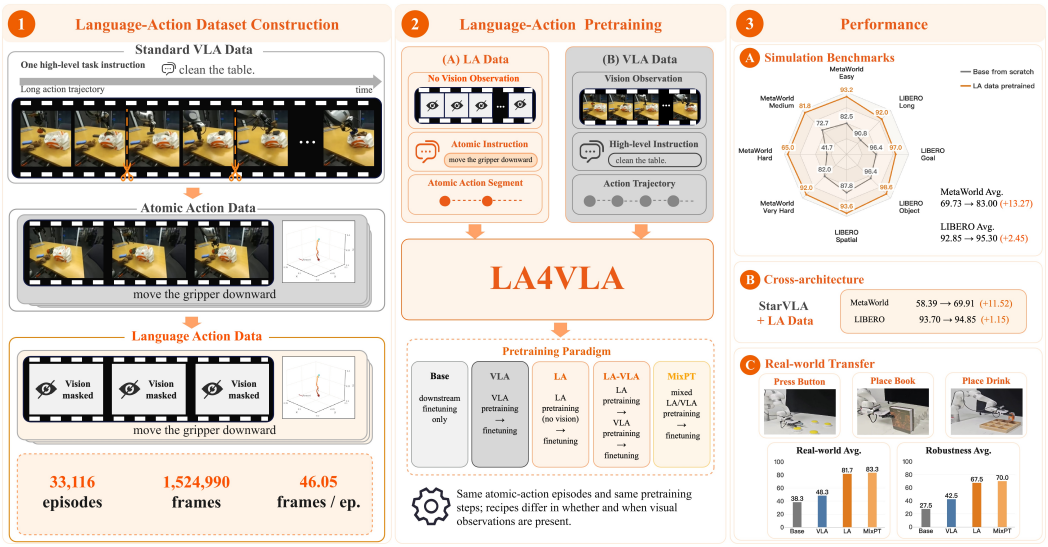


Figure 1: Overview of the LA4VLA framework. The left panel illustrates the construction of Language-Action data by reorganizing VLA demonstrations into atomic instruction-action segments. The middle panel summarizes the LA4VLA pretraining paradigm, which studies how LA supervision can be used alone or combined with standard VLA pretraining. The right panel presents representative performance across simulation, architectures, real-world transfer, and robustness evaluations.

learning toward visual signals and dilute the supervision for learning how language conditions actions. Consequently, a policy may appear language-conditioned, while its internal representations remain dominated by visual-action associations. This can lead to visual shortcuts that weaken the influence of language intent, leaving language-action grounding underdeveloped (Xu et al., 2025; Fang et al., 2026b). Such entanglement becomes especially problematic under visual variations, such as changes in viewpoint, background, lighting, object appearance, or scene layout, where robust manipulation requires actions to remain conditioned on the instruction rather than on spurious visual correlations.

This motivates us to rethink VLA pretraining through one key question: ***Can language-action grounding be decoupled from visually dominated VLA learning?*** We hypothesize that explicitly removing visual observations during pretraining can prevent the policy from exploiting visual shortcuts and encourage it to focus on the intrinsic relationship between language and action. To this end, we explore vision-agnostic language-action pretraining, where the policy predicts actions from language instructions and proprioceptive states without visual observations. The objective is not intended to replace visual grounding, since object localization and scene-specific target selection still require vision. Instead, it encourages the policy to learn vision-independent action knowledge, such as action dynamics, temporal consistency, and language-conditioned motion patterns.

Building on this idea, we propose **LA4VLA**, a language-action pretraining framework that constructs explicit Language-Action (LA) supervision and integrates it into VLA learning. Specifically, LA4VLA decomposes complete demonstration trajectories into short atomic action segments and associates each segment with a corresponding low-level action description. The resulting LA pairs enable the policy to learn how language conditions action execution without relying on visual observations. These atomic actions, such as moving, grasping, lifting, transporting, pressing, and rotating, capture reusable manipulation skills shared across tasks and scenes. Based on this trajectory decomposition, we build **LA-33K**, a dataset containing 33K LA pairs derived entirely from existing expert trajectories (Khazatsky et al., 2024), without requiring additional robot data collection. Rather than merely scaling the amount of training data, LA4VLA reorganizes existing demonstrations to expose explicit language-action supervision that is often implicit or diluted in standard VLA training. Pretraining on these vision-agnostic LA pairs enables the policy to acquire reusable language-conditioned action priors that can strengthen downstream VLA learning.

We systematically evaluate the effectiveness of LA pretraining by developing **LA4VLA-1B**, a lightweight 1B-parameter VLA model, and testing it across several simulation benchmarks and real-world robot manipulation tasks. To examine how language-action supervision can be integrated

into VLA policy learning, we study a series of pretraining paradigms, including pure LA pretraining, sequential LA-to-VLA pretraining, and mixed LA-VLA pretraining. Consistent results across LA4VLA-1B and StarVLA (Community, 2026) show that pretraining on LA data alone yields stronger downstream improvements than pretraining with VLA data alone, while combining LA and VLA data leads to further gains. These findings demonstrate that language-action supervision provides an effective and transferable training signal for downstream VLA policies and is complementary to existing VLA learning pipelines. In particular, LA4VLA-1B achieves average success rates of 87.53% on MetaWorld (Yu et al., 2020), 96.28% on LIBERO (Liu et al., 2023), 70.40% on the LIBERO-Plus suite (Fei et al., 2025), and 83.3% on real-world manipulation tasks, improving over the no-pretraining baselines by 17.80, 3.43, 11.62, and 45.0 percentage points, respectively. Beyond average task success, our analysis further shows that LA pretraining improves robustness under visual perturbations, supporting the hypothesis that language-conditioned action priors can reduce over-reliance on visual shortcuts. Figure 1 provides an overview of the LA4VLA framework, including LA data construction, the pretraining paradigm, and representative performance across simulation benchmarks, architectures, and real-world manipulation tasks. Overall, our main contributions are summarized as follows:

- We propose LA4VLA, a language-action pretraining framework that decouples language-conditioned action learning from visually grounded VLA training, enabling policies to learn reusable action priors that generalize across scenes and tasks without over-relying on visual cues.
- We construct Language-Action (LA) data from existing expert demonstrations by decomposing full trajectories into atomic action segments and pairing each segment with a low-level action description. This results in LA-33K, a dataset of 33K vision-agnostic LA episodes created without additional robot data collection.
- We show that LA pretraining consistently enhances VLA policy performance across simulation benchmarks, model architectures, and real-world manipulation tasks, improves robustness under visual perturbations, and yields further gains when combined with standard VLA data.

2 Related Work

Vision-language-action models and robot pretraining. Early robot transformer models showed that large-scale behavior cloning supports multi-task manipulation from visual observations and language instructions (Brohan et al., 2022; Zitkovich et al., 2023; O'Neill et al., 2024), and recent VLA models scale this paradigm with stronger vision-language backbones, larger robot datasets, and more expressive action heads (Kim et al., 2024; Black et al., 2024; Bjorck et al., 2025; Shukor et al., 2025; Lin et al., 2026b; Community, 2026; Wang et al., 2026b). Despite differences in architectures, action representations, and training recipes, they share the same supervision format: the policy is pretrained on demonstrations that jointly pair visual observations, language instructions, and action trajectories, which couples visual grounding and action learning from the beginning. Improvements such as better action tokenization, diffusion or flow-matching action heads, larger multi-robot datasets, and cross-embodiment conditioning (Pertsch et al., 2025; Black et al., 2024; Lin et al., 2026a; Wang et al., 2026b; Jia et al., 2026; Du et al., 2026; Liu et al., 2025) still operate within this joint training format, so the contribution of language-action supervision itself remains difficult to isolate. It remains unclear whether the policy learns reusable language-conditioned action knowledge or mainly uses language as a weak condition alongside dense visual observations. In contrast, our work explicitly separates language-action learning from visual grounding during pretraining, allowing us to study how vision-agnostic action supervision can benefit VLA policies.

Language and action supervision for robot learning. Language-conditioned imitation learning trains policies to execute natural-language commands from visual observations (Jang et al., 2022; Lynch et al., 2023; Shridhar et al., 2022), and other systems use language models to decompose tasks, generate plans, or provide reasoning traces (Ahn et al., 2022; Liang et al., 2023; Zhao et al., 2025; Shi et al., 2025; Li et al., 2026; Mon-Williams et al., 2025). While these approaches use language to specify tasks or guide planning, more recent work has introduced language-to-action signals more explicitly. LAP represents low-level robot motions as natural-language action strings and trains a VLM backbone to predict them (Zha et al., 2026), while Qwen-VLA includes a text-to-action stage that trains an action decoder from language and embodiment prompts (Wang et al., 2026b). These studies demonstrate that language can provide more direct supervision for action learning, but do

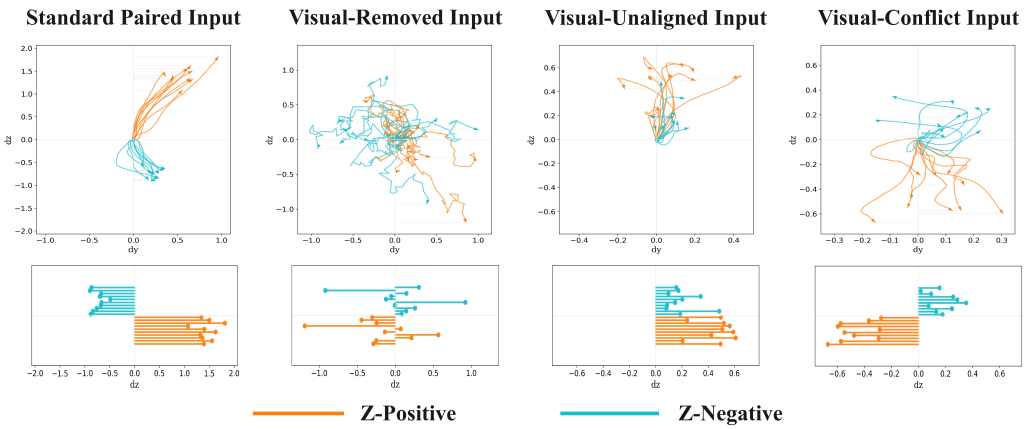


Figure 2: Qualitative visualization of instruction-conditioned direction following under different visual input configurations. The top and bottom rows show predicted trajectories and endpoint displacements, respectively, projected onto the plane containing the tested motion axis.

not investigate vision-agnostic language-action supervision as a pretraining signal for VLA policy learning. In contrast, our work pairs atomic instructions with their corresponding action trajectories, removes visual observations during LA pretraining, and systematically studies how this supervision can be used alone or combined with standard VLA pretraining.

3 Probing Instruction Following in VLAs

To examine whether a VLA policy predicts actions by following the instruction or by exploiting the paired visual observation, we design a controlled diagnostic study. We keep the language instruction fixed while removing, mismatching, or conflicting the visual input, and examine whether the predicted action direction changes. If a policy has learned robust language-action grounding, fixing the instruction should keep the predicted motion aligned with the commanded direction regardless of the visual input. We focus on atomic direction-following actions such as "lower the object downward" and "transport the object to the left", where the expected direction is measurable.

3.1 Diagnostic Evaluation

Policy and task setup. We use a VLA policy built on InternVL3-1B (Zhu et al., 2025) with a flow matching action head. The policy is trained under the standard VLA format, which pairs visual observations, language instructions, robot states, and action trajectories. The only difference from standard VLA training is that the policy is trained and evaluated on atomic direction-following trajectories, where the end effector moves along a specified Cartesian direction. We construct evaluation cases from instruction pairs that specify opposite directions along the same Cartesian axis.

Evaluated input conditions. We evaluate four visual-input configurations while keeping the target instruction L and robot state fixed: 1) *Standard paired input* (L, V^+), where V^+ is the observation originally paired with the instruction; 2) *Visual-removed input* $(L, \varnothing)$, where the observation is masked out; 3) *Visual-unaligned input* $(L, \tilde{V})$, where the observation is replaced with a same-scene image not paired with the current instruction; and 4) *Visual-conflict input* (L, V^-), where the observation comes from a trajectory associated with the opposite motion direction. Since the instruction and robot state remain fixed across all conditions, changes in the predicted action direction can be attributed to the visual input, allowing us to assess whether the policy follows the language instruction or relies on the paired visual observation.

Metrics. We use four metrics to quantify whether the predicted motion follows the instruction. The first two measure directional alignment: *Directional Alignment Rate* (DAR) is the fraction of trials whose endpoint displacement falls in the instruction-aligned half-space, and *Direction Consistency Score* (DCS) is the cosine similarity between the predicted and ground-truth endpoint displacements. The other two measure whether instructions specifying different target directions produce separable motion patterns: *Separability Ratio* (SR) compares the average distance across different instruction

Table 1: Instruction-following diagnostic under different visual input configurations. Metrics are computed over 100 direction-following cases for each evaluation condition.

Condition	Inputs	DAR ↑	DCS ↑	SR ↑	SS ↑
Standard paired input	(L, V^+)	0.98	0.95	2.35	0.55
Visual-removed input	$(L, \varnothing)$	0.63	0.16	1.03	0.04
Visual-unaligned input	$(L, \tilde{V})$	0.66	0.37	1.13	0.05
Visual-conflict input	(L, V^-)	0.35	0.03	1.03	-0.04

directions with the average distance within the same instruction direction, while *Silhouette Score* (SS) is a normalized clustering measure of whether each endpoint displacement is closer to its own instruction-direction cluster than to other clusters. Formal definitions are in Appendix E.

3.2 Diagnostic Results

Qualitative visualization. Figure 2 visualizes the predicted trajectories and endpoint displacements for a representative pair of instructions specifying opposite motion directions, "move upward to approach the target" and "move downward to approach the target", both projected onto the plane containing the tested motion axis. Under the *Standard paired input*, the model produces clearly separated trajectories for the two instructions, with endpoint projections falling on the instruction-aligned sides of the tested axis. However, the following visual interventions reveal that this alignment is fragile and not robustly grounded in the language instruction itself. With the *Visual-removed input*, the trajectories become scattered and the two instruction directions are poorly separated. With the *Visual-unaligned input*, opposite instructions no longer produce clearly separable endpoint distributions, indicating that the policy cannot reliably distinguish actions based on language once the paired visual observation is replaced. Under the *Visual-conflict input*, the substituted visual observation is associated with the opposite motion direction. In this case, the endpoint projections shift away from the instruction-aligned direction and become biased toward the direction implied by the substituted visual observation. These results show that the policy is highly sensitive to visual input and may deviate from the language instruction when the paired visual cue changes.

Quantitative analysis. Table 1 reports average results over 100 direction-following cases per condition. Under the *Standard paired input*, the model reaches DAR 0.98 and DCS 0.95, while the high SR (2.35) and SS (0.55) indicate clear endpoint separation between opposite instructions. However, this apparent instruction following breaks under visual intervention. With the *Visual-removed input*, DCS drops to 0.16 and SS to nearly zero, indicating that opposite instructions no longer induce clearly separable actions without the paired image. With the *Visual-unaligned input*, DCS remains low at 0.37 and SS is 0.05, showing that replacing the paired image with an unaligned same-scene observation is sufficient to disrupt instruction-conditioned action prediction. Under the *Visual-conflict input*, DAR drops to 0.35, below the 0.5 expected under random directional choice, and DCS falls to 0.03. This means that predictions fall in the opposite half-space more often than in the instruction-aligned half-space, indicating that the policy is not merely uncertain but can be biased away from the instruction by a conflicting visual observation. These quantitative results are consistent with the qualitative visualizations in Figure 2: standard paired inputs can create apparent instruction following, but once the paired visual cue is removed, unaligned, or placed in conflict, the policy fails to reliably distinguish actions according to language. Since the instruction is fixed across these conditions, this behavior directly supports our diagnostic assumption that changes in predicted motion reflect the policy's dependence on the substituted visual input. Together, the qualitative and quantitative results suggest that standard VLA training does not robustly ground action execution in language, motivating more direct language-action supervision without paired visual observations.

4 LA4VLA: Language-Action Dataset Construction

4.1 Overview

The diagnostic analysis above shows that the instruction-following behavior of VLA policies can become unstable under changes in visual input, suggesting that language-action supervision should be exposed more directly rather than always being tied to paired visual observations. To this end,

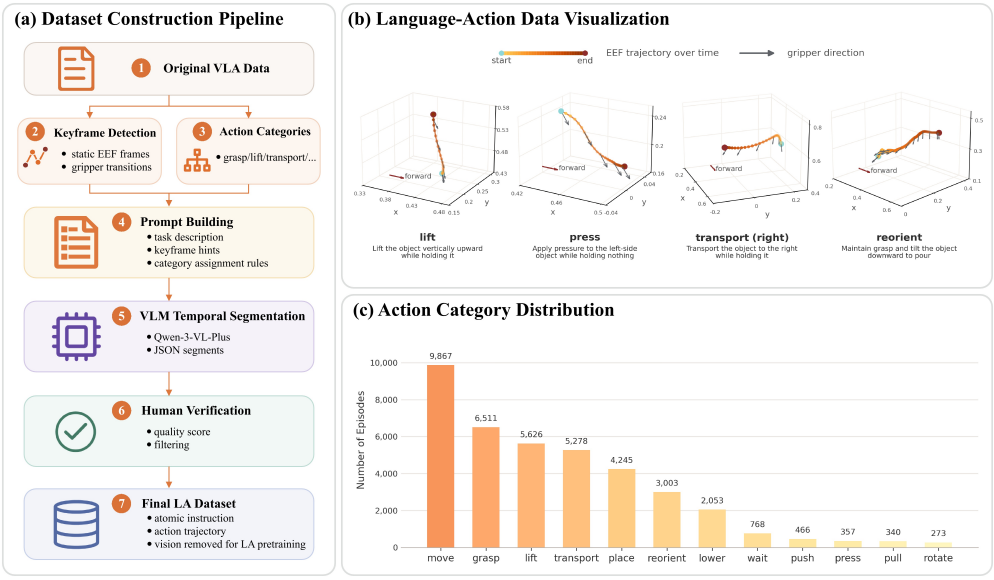


Figure 3: Overview of the Language-Action dataset. The left panel summarizes the construction pipeline, which converts standard vision-language-action demonstrations into vision-agnostic atomic language-action episodes. The top-right panel visualizes representative Language-Action data examples using end-effector trajectories and gripper-direction arrows. The bottom-right panel reports the action-category distribution of the LA-33K dataset.

we propose the LA4VLA framework for vision-agnostic language-action pretraining, where models learn from language instructions, proprioceptive states, and action trajectories without visual inputs.

To convert standard VLA demonstrations into LA episodes, LA4VLA introduces a construction pipeline that segments long demonstration trajectories into short atomic action intervals and annotates each interval with a low-level instruction. We apply this pipeline to a large-scale robot manipulation dataset (Khazatsky et al., 2024), resulting in LA-33K, a dataset of 33K Language-Action episodes constructed without additional robot data collection. By reorganizing standard VLA demonstrations into explicit instruction-action pairs, LA-33K exposes language-action supervision that is often implicit or diluted in full demonstration trajectories. Figure 3 summarizes the construction pipeline, visualized examples of LA data, and the action-category distribution of LA-33K.

4.2 Dataset Construction Pipeline

As shown in Figure 3(a), LA4VLA converts standard VLA demonstrations into vision-agnostic atomic Language-Action episodes through a construction pipeline. The retained segments contain an atomic instruction, robot states, and the corresponding action trajectory, with visual observations removed for LA pretraining.

Starting from the *Original VLA Data*, we first perform *Keyframe Detection* to identify static EEF frames and gripper transitions, and define *Action Categories* for candidate segments. Together with the original task description, these cues are used for *Prompt Building*. Qwen-3-VL-Plus (Bai et al., 2025) then performs *VLM Temporal Segmentation* to produce candidate JSON segments. During *Human Verification*, annotators assign quality scores and filter segments whose temporal boundaries, action labels, and language descriptions are not mutually consistent. The retained segments form the *Final LA Dataset*. Figure 3(b) visualizes representative LA episodes, where end-effector trajectories and gripper directions illustrate how atomic instructions correspond to action trajectories. Detailed taxonomy design and implementation settings are provided in Appendix A.

4.3 Dataset Statistics

Applying the construction pipeline to DROID (Khazatsky et al., 2024) yields LA-33K. Starting from 9,560 original VLA episodes containing 2,751,615 frames, the pipeline produces 33,116 final LA

episodes containing 1,524,990 frames, reducing the average episode length from 287.83 to 46.05 frames. Figure 3(c) shows that the final dataset covers 12 atomic action categories across object manipulation, contact interaction, free-space motion, and rotation. Detailed verification procedures and processing statistics are provided in Appendices A.6 and A.7, respectively.

5 LA4VLA: LANGUAGE-ACTION PRETRAINING

5.1 EXPERIMENTAL DESIGNS

This section systematically evaluates language-action pretraining as a VLA learning paradigm, examining its downstream performance, integration with VLA pretraining, transferability, robustness, and effects on instruction-conditioned action prediction.

Pretraining paradigms. We study a pretraining paradigm built around vision-agnostic language-action supervision for VLA policy learning. We use two paired pretraining data formats derived from the same atomic episodes. The first is LA-33K, where visual observations are removed and each episode contains a vision-agnostic instruction, robot states, and the corresponding action trajectory. The second is LA-33K-V, a VLA-format counterpart of LA-33K that restores the corresponding original visual observations for each atomic episode. **Base** directly finetunes the policy on downstream task data without pretraining. **LA** pretrains on LA-33K without visual input before the same downstream finetuning stage. **VLA** pretrains on LA-33K-V with visual observations enabled, and then finetunes on the downstream task. **LA-VLA** first performs LA pretraining on LA-33K, then performs VLA pretraining on LA-33K-V, and finally finetunes on downstream task data. **MixPT** jointly uses LA-33K and LA-33K-V within a single mixed pretraining stage, followed by the same downstream finetuning.

Experimental setup. We propose **LA4VLA-1B**, a 1B-parameter VLA model built on InternVL3-1B (Zhu et al., 2025) with a flow matching action head (Appendix B). We pretrain the model under the different paradigms described above. For fair comparisons within each experiment group, we keep the model architecture, training budget, optimization hyperparameters, downstream data, and finetuning protocol fixed, changing only the pretraining paradigm under study.

Questions and findings. The experiments address six questions:

- **Does language-action pretraining improve downstream performance?** Yes, LA-pretrained policies consistently outperform scratch-trained or VLA-pretrained policies (Section 5.2).
- **How should LA supervision be combined with VLA pretraining?** Adding visual grounding after (LA-VLA) or together with (MixPT) language-action pretraining further strengthens downstream policies (Section 5.3).
- **Does the benefit generalize across VLA architectures?** Yes, LA pretraining transfers to another VLA architecture with consistent gains (Section 5.4).
- **Does the benefit extend to real robots?** Yes, LA pretraining leads to stronger real-world manipulation capability (Section 5.5).
- **Is the benefit robust to visual perturbations?** Yes, LA-pretrained policies improve robustness under LIBERO-Plus scene and visual perturbations, and degrade less under real-world image noise (Sections 5.2 and 5.5).
- **Do the predicted actions follow the language instruction?** Yes, qualitative trajectories and quantitative direction-following metrics show that LA-pretrained policies produce instruction-aligned actions without visual input (Section 5.6).

5.2 EFFECT OF LANGUAGE-ACTION PRETRAINING

We first evaluate whether language-action pretraining improves downstream manipulation performance (protocols in Appendix C). Table 2 reports results on MetaWorld (Yu et al., 2020), LIBERO (Liu et al., 2023), and the LIBERO-Plus (Fei et al., 2025) suite, which evaluates LIBERO-10 under layout, background, noise, and camera variations. Across these benchmarks, LA pretraining improves average success over the Base policy by 13.27, 2.45, and 10.24 percentage points, respectively, and over VLA pretraining by 3.22, 0.90, and 10.01 percentage points. These consistent gains show that, under the matched atomic-action pretraining setting, vision-agnostic language-action supervision provides a more effective pretraining signal than standard VLA pretraining for downstream

Table 2: Downstream performance under different pretraining schemes. The best and second-best values in each numeric column are marked in **bold** and underline, respectively. Reference results are taken from original papers or official reproductions reported in existing work.

Model	Pretrain	Success Rate (%)					Gain
MetaWorld		**Easy**	**Medium**	**Hard**	**Very Hard**	**Avg.**	
Evo-1 (Lin et al., 2026b)	No	89.2	76.8	77.2	79.2	80.6	–
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	Yes	86.1	78.2	80.0	82.0	81.6	–
Evo-depth (Lin et al., 2026a)	No	83.1	84.7	**87.3**	82.4	84.4	–
StarVLA (Community, 2026)	No	85.7	60.0	40.6	47.3	58.39	+0.00
StarVLA (Community, 2026)	LA	85.1	62.2	55.0	77.3	69.91	+11.52
LA4VLA-1B	No	82.5	72.7	41.7	82.0	69.73	+0.00
LA4VLA-1B	VLA	88.6	68.2	68.3	94.0	79.78	+10.05
LA4VLA-1B	LA	**93.2**	81.8	65.0	92.0	83.00	+13.27
LA4VLA-1B	LA-VLA	91.4	90.9	66.7	98.0	86.75	+17.02
LA4VLA-1B	MixPT	88.9	**94.5**	66.7	**100.0**	**87.53**	**+17.80**
LIBERO		**Spatial**	**Object**	**Goal**	**Long**	**Avg.**	
π_0 (Black et al., 2024)	VLA	96.8	98.8	95.8	85.2	94.2	–
Evo-1 (Lin et al., 2026b)	No	92.7	97.7	96.3	92.3	94.8	–
Evo-depth (Lin et al., 2026a)	No	95.6	99.2	95.6	91.3	95.4	–
StarVLA (Community, 2026)	No	96.4	98.4	95.8	84.2	93.70	+0.00
StarVLA (Community, 2026)	LA	**97.4**	99.0	96.0	87.0	94.85	+1.15
LA4VLA-1B	No	87.8	96.4	96.4	90.8	92.85	+0.00
LA4VLA-1B	VLA	93.2	**99.4**	96.8	88.2	94.40	+1.55
LA4VLA-1B	LA	93.6	98.6	97.0	92.0	95.30	+2.45
LA4VLA-1B	MixPT	93.4	98.2	98.0	**93.4**	95.75	+2.90
LA4VLA-1B	LA-VLA	95.8	97.4	**98.5**	**93.4**	**96.28**	**+3.43**
LIBERO-Plus		**Layout**	**Background**	**Noise**	**Camera**	**Avg.**	
OpenVLA (Kim et al., 2024)	VLA	28.3	19.4	17.6	0.0	16.33	–
UniVLA (Bu et al., 2025)	VLA	16.4	74.4	25.4	1.9	29.53	–
π_0 (Black et al., 2024)	VLA	69.6	69.5	64.4	3.8	51.83	–
LA4VLA-1B	No	59.94	80.97	63.92	30.31	58.78	+0.00
LA4VLA-1B	VLA	71.25	79.52	68.24	17.02	59.01	+0.22
LA4VLA-1B	LA	81.41	84.08	**78.62**	31.98	69.02	+10.24
LA4VLA-1B	MixPT	**82.71**	**88.13**	77.42	**33.35**	**70.40**	**+11.62**

VLA policies. We next examine how language-action supervision should be integrated with VLA pretraining.

5.3 Integrating Language-Action Supervision with VLA Pretraining

Table 2 further compares two integration paradigms: LA-VLA (sequential LA then VLA pretraining) and MixPT (mixed LA and VLA in one stage). On MetaWorld, LA-VLA improves over LA-only from 83.00% to 86.75%, and MixPT reaches 87.53%. On LIBERO, LA-VLA reaches 96.28% and MixPT 95.75%, both above LA-only at 95.30%. On the LIBERO-Plus suite, MixPT further improves over LA-only from 69.02% to 70.40%, while remaining far above VLA pretraining at 59.01%. These results show that language-action supervision is complementary to VLA pretraining, and both staged and mixed integration are effective where evaluated.

5.4 Generalization to Different VLA Architectures

We apply the same LA pretraining protocol to StarVLA (Community, 2026), another VLA architecture, and compare it against StarVLA without pretraining under the same downstream finetuning protocol. As shown in Table 2, LA pretraining increases the average success rate by 11.52 percentage points on MetaWorld and 1.15 percentage points on LIBERO, demonstrating that language-action supervision transfers across VLA architectures.

5.5 Real-World Manipulation Performance

Real-world evaluation. We evaluate real-world manipulation on three language-conditioned task families: *Press Button*, *Place Book*, and *Place Drink*. Together, they cover 11 language-specified

Table 3: Real-world manipulation performance under different pretraining schemes. Success rates are reported in percent. Each task is evaluated with 20 trials per policy.

Model	Pretrain	Press Button	Place Book	Place Drink	Avg.	Gain
LA4VLA-1B	No	60.0	15.0	40.0	38.3	+0.0
LA4VLA-1B	VLA	50.0	40.0	55.0	48.3	+10.0
LA4VLA-1B	LA	**85.0**	65.0	**95.0**	81.7	+43.4
LA4VLA-1B	MixPT	75.0	**85.0**	90.0	**83.3**	**+45.0**

Table 4: Robustness under Gaussian image noise. Success rates (%) with 20 trials per task.

Model	Pretrain	Press Button	Place Drink	Avg.	Gain
LA4VLA-1B	No	40.0	15.0	27.5	+0.0
LA4VLA-1B	VLA	20.0	65.0	42.5	+15.0
LA4VLA-1B	LA	55.0	**80.0**	67.5	+40.0
LA4VLA-1B	MixPT	**65.0**	75.0	**70.0**	**+42.5**

target goals: four button locations, three shelf positions, and four grid cells, respectively. Within each task family, the initial scene and visual observations are shared across target goals, so the visual input alone cannot disambiguate the target and the instruction determines which goal to execute. We collect 100 teleoperated demonstrations per task family (300 in total) and finetune a multi-task policy for each pretraining scheme. Representative rollouts and experimental setup are provided in Appendix D.

Table 3 shows that LA pretraining substantially improves real-world manipulation across all three task families. Compared with VLA pretraining, LA pretraining performs better on *Press Button*, *Place Book*, and *Place Drink*, while both LA and MixPT improve average success by more than 43 percentage points over direct finetuning. MixPT achieves the highest average success, indicating that language-action supervision provides an effective basis for distinguishing language-specified goals that share the same visual scene, and can be further complemented by visual grounding. This demonstrates that the benefit of LA pretraining extends beyond simulation to real-world manipulation.

Robustness to visual perturbations. We further evaluate robustness to visual perturbations by adding Gaussian noise to the camera observations for *Press Button* and *Place Drink*. Evaluation details are provided in Appendix D. As shown in Table 4, LA pretraining maintains a substantial advantage over both direct finetuning and VLA pretraining under noisy visual observations. MixPT achieves the best average success rate, improving from 27.5% without pretraining to 70.0%. Together with the clean-observation results, these findings show that the benefit of LA supervision persists when the paired visual input is degraded, rather than being limited to the original visual conditions.

5.6 Instruction-Following Behavior and Representation Analysis

Following the diagnostic study in Section 3, we use qualitative trajectory visualizations and quantitative direction-following metrics to examine whether LA pretraining improves language-action grounding without visual input. Figure 9 in Appendix F visualizes LA-pretrained trajectories that follow the language instruction, together with a t-SNE visualization showing more instruction-aligned action-decoding representations. This qualitative evidence is also supported by the quantitative results in Table 15: the LA-pretrained policy achieves a DCS of 0.85 and a DAR of 0.99, confirming improved instruction-conditioned action prediction after LA pretraining.

6 Conclusion

We presented LA4VLA, a language-action pretraining framework that constructs vision-agnostic atomic instruction-action pairs from existing robot demonstrations and uses them to learn language-conditioned action priors before downstream VLA learning. This yields LA-33K, a 33K-episode Language-Action dataset derived without additional data collection. Across simulation benchmarks and real-world manipulation tasks, LA pretraining consistently outperforms VLA-only pretraining under comparable settings, transfers across VLA architectures, remains especially beneficial under visual perturbations, and provides further gains when combined with standard VLA training. These results highlight the importance of making language-action supervision explicit rather than leaving it entangled within joint vision-language-action training.

Reproducibility statement

We provide detailed descriptions of the LA dataset construction and verification process in Appendix A, model and training configurations in Appendix B, simulation benchmarks and evaluation protocols in Appendix C, real-world experimental protocols in Appendix D, and formal metric definitions in Appendix E. We will release the code, datasets, and pretrained model weights to support reproduction and future research.

AI use statement

Generative AI tools were used only for language editing and writing refinement. They were not used to formulate the research ideas, design the methodology or experiments, generate or analyze experimental data, or draw scientific conclusions. The authors reviewed all AI-assisted edits and take full responsibility for the content of this paper.

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.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

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.

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

StarVLA Community. Starvla: A lego-like codebase for vision-language-action model developing. *arXiv preprint arXiv:2604.05014*, 2026.

Yuxin Du, Tao Lin, Zile Zhong, Runting Li, Xiyao Chen, Jiting Liu, Chenglin Liu, Ying-Cong Chen, Yuqian Fu, and Bo Zhao. Focusable monocular depth estimation. *arXiv preprint arXiv:2605.11756*, 2026.

Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, et al. Molmoact2: Action reasoning models for real-world deployment. *arXiv preprint arXiv:2605.02881*, 2026a.

Yu Fang, Yuchun Feng, Dong Jing, Jiaqi Liu, Yue Yang, Zhenyu Wei, Daniel Szafir, and Mingyu Ding. When vision overrides language: Evaluating and mitigating counterfactual failures in vlas. *arXiv preprint arXiv:2602.17659*, 2026b.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *conference on Robot Learning*, pp. 991–1002. PMLR, 2022.

Jindou Jia, Gen Li, Xiangyu Chen, Tuo An, Yuxuan Hu, Jingliang Li, Xinying Guo, and Jianfei Yang. A2a: Action-to-action flow matching policy. In *Proceedings of Robotics: Science and Systems*, 2026.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

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.

Gen Li, Bo Zhao, Jianfei Yang, and Laura Sevilla-Lara. Mask2iv: Interaction-centric video generation via mask trajectories. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 6091–6099, 2026.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International conference on robotics and automation (ICRA)*, pp. 9493–9500. IEEE, 2023.

Tao Lin, Gen Li, Yilei Zhong, Yanwen Zou, Yuxin Du, Jiting Liu, Encheng Gu, and Bo Zhao. Evo-0: Vision-language-action model with implicit spatial understanding. *arXiv preprint arXiv:2507.00416*, 2025.

Tao Lin, Yuxin Du, Jiting Liu, Nuobei Zhu, Yunhe Li, Yuqian Fu, Yinxinyu Chen, Hongyi Cai, Zewei Ye, Bing Cheng, et al. Evo-depth: A lightweight depth-enhanced vision-language-action model. *arXiv preprint arXiv:2605.14950*, 2026a.

Tao Lin, Yilei Zhong, Yuxin Du, Jingjing Zhang, Jiting Liu, Yinxinyu Chen, Encheng Gu, Ziyan Liu, Hongyi Cai, Yanwen Zou, et al. Evo-1: Lightweight vision-language-action model with preserved semantic alignment. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13397–13406, 2026b.

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.

Ziyan Liu, Yeqiu Chen, Hongyi Cai, Tao Lin, Shuo Yang, Zheng Liu, and Bo Zhao. Vla-pruner: Temporal-aware dual-level visual token pruning for efficient vision-language-action inference. *arXiv preprint arXiv:2511.16449*, 2025.

Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. Interactive language: Talking to robots in real time. *IEEE Robotics and Automation Letters*, 2023.

Ruaridh Mon-Williams, Gen Li, Ran Long, Wenqian Du, and Christopher G Lucas. Embodied large language models enable robots to complete complex tasks in unpredictable environments. *Nature Machine Intelligence*, 7(4):592–601, 2025.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

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.

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.

Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, et al. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. *arXiv preprint arXiv:2502.19417*, 2025.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on robot learning*, pp. 894–906. PMLR, 2022.

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

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.

Kuanning Wang, Ke Fan, Chenhao Qiu, Zeyu Shangguan, Yuqian Fu, Yanwei Fu, Daniel Seita, and Xiangyang Xue. Oflow: Injecting object-aware temporal flow matching for robust robotic manipulation. *arXiv preprint arXiv:2604.17876*, 2026a.

Qiuyue Wang, Mingsheng Li, Jian Guan, Jinhui Ye, Sicheng Xie, Yitao Liu, Junhao Chen, Zhixuan Liang, Jie Zhang, Xintong Hu, et al. Qwen-vla: Unifying vision-language-action modeling across tasks, environments, and robot embodiments. *arXiv preprint arXiv:2605.30280*, 2026b.

Runze Wang, Yuqian Fu, Yu Li, Tao Lin, Tianwen Qian, Mohamed Elhoseiny, Bo Zhao, Yanwei Fu, Yu-Gang Jiang, and Xiangyang Xue. Afford-vla: Action-aligned visual planning via internalized affordance. *arXiv preprint arXiv:2605.24203*, 2026c.

Kechun Xu, Zhenjie Zhu, Anzhe Chen, Shuqi Zhao, Qing Huang, Yifei Yang, Haojian Lu, Rong Xiong, Masayoshi Tomizuka, and Yue Wang. Seeing to act, prompting to specify: A bayesian factorization of vision language action policy. *arXiv preprint arXiv:2512.11218*, 2025.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Lihan Zha, Asher J Hancock, Mingtong Zhang, Tenny Yin, Yixuan Huang, Dhruv Shah, Allen Z Ren, and Anirudha Majumdar. Lap: Language-action pre-training enables zero-shot cross-embodiment transfer. *arXiv preprint arXiv:2602.10556*, 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. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 1702–1713, 2025.

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

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.

Kuangji Zuo, Gen Li, Bofan Lyu, Yanshuo Lu, Boyu Ma, Shijia Han, Xinyu Zhou, Xichen Yuan, Chuhao Zhou, Jiaqi Bai, et al. Gaze2act: Gaze-conditioned vision-language-action policies for interactive robot manipulation. *arXiv preprint arXiv:2605.30282*, 2026.

APPENDIX

A LA DATASET CONSTRUCTION DETAILS

This appendix complements the dataset construction pipeline described in Section 4. The main text presents the overall flow from original VLA data to the final LA dataset. Here we provide the implementation details for keyframe detection, atomic-action design criteria, primitive definitions, prompt construction, VLM-based temporal segmentation, human verification, and dataset statistics.

A.1 KEYFRAME DETECTION

Purpose. Keyframes provide temporal cues that can assist the segmentation model in identifying candidate atomic-action phases. We use them as auxiliary hints in the segmentation prompt: gripper transition frames often correspond to grasp or release/place phases, while static frames indicate near-stationary intervals that commonly occur around alignment, grasping, placing, or phase transitions.

Static keyframes. Static keyframes mark short intervals in which the end-effector motion remains low. We use the 6D Cartesian EEF pose $\mathbf{s}_t^{\text{eef}} = (x_t, y_t, z_t, \text{roll}_t, \text{pitch}_t, \text{yaw}_t)$ and treat the scalar gripper position g_t separately. We first compute adjacent-frame pose differences,

$$\Delta \mathbf{s}_t = \mathbf{s}_{t+1}^{\text{eef}} - \mathbf{s}_t^{\text{eef}}.$$

Each dimension of $\Delta \mathbf{s}_t$ is smoothed with a Savitzky–Golay filter using window length 5 and polynomial order 3. The smoothed differences are then normalized per dimension by their episode-wise maximum absolute value, with scale floors of 0.02 for translation and 0.05 for rotation. We aggregate the six normalized dimensions using the mean absolute value to obtain a scalar motion magnitude m_t. The static threshold is

$$\theta_s = \max(0.3\,\text{median}(m),\, 0.01)\,.$$

A frame is retained as a static candidate when at least 80% of the frames within a radius-3 temporal window satisfy $m_t \le \theta_s$. Strictly consecutive candidates are merged into static regions. Regions containing at least 8 frames are retained as static intervals, while shorter regions are represented by their midpoint frames.

Gripper transition keyframes. Gripper transitions are detected directly from adjacent-frame differences $\Delta g_t = g_{t+1} - g_t$ of the raw scalar gripper position. We smooth Δg_t with a centered moving-average window of length 3. Under the dataset convention that increasing gripper position indicates closing, values above 0.01 are labeled as gripper-close and values below -0.01 as gripper-open. Same-direction transition runs separated by at most 5 frames are merged, runs shorter than 4 difference frames are discarded, and the retained runs are converted into gripper-transition intervals.

Algorithm. The resulting detector combines low-motion intervals and gripper-state transitions:

Algorithm 1 Keyframe detection from robot state trajectories

Require: EEF pose trajectory $\mathbf{s}_{1:T}^{\text{eef}}$, gripper trajectory $g_{1:T}$, and timestamps $\tau_{1:T}$
Ensure: Sorted static and gripper-transition cues $\mathcal{K}$

1: Compute EEF differences $\Delta \mathbf{s}_t = \mathbf{s}_{t+1}^{\text{eef}} - \mathbf{s}_t^{\text{eef}}$
2: Smooth each dimension with a Savitzky–Golay filter (window 5, order 3)
3: Normalize each dimension using its maximum absolute value and configured scale floor
4: Compute m_t as the mean absolute normalized EEF difference
5: Set $\theta_s = \max(0.3\,\text{median}(m), 0.01)$
6: Retain frames whose radius-3 window is at least 80% static
7: Merge consecutive frames; retain long regions as intervals and short regions as midpoint keyframes
8: Compute $\Delta g_t = g_{t+1} - g_t$ and smooth it with a length-3 moving average
9: Label $\Delta g_t > 0.01$ as close and $\Delta g_t < -0.01$ as open
10: Merge same-direction runs within 5 frames and discard runs shorter than 4 difference frames
11: Convert retained events to temporal cues and sort them into $\mathcal{K}$
12: **return** $\mathcal{K}$

Prompt formatting. Detected keyframes are inserted into the VLM prompt as timestamped state events, for example:

```
- 1.20s:  static
- 1.80s:  gripper_close
- 3.40s:  gripper_open
```

These events are provided as temporal hints for segmentation rather than hard boundaries, allowing the VLM to jointly use robot-state cues and visual evidence when determining segment boundaries.

A.2 Atomic-Action Design Criteria

Design criteria. Table 5 summarizes the high-level atomic-action taxonomy used in the main text. The taxonomy is designed to define short segments that support language-action supervision and is guided by three criteria:

- **Local temporal boundary.** Each segment should cover a short contiguous time interval and avoid spanning multiple task phases.
- **Observable physical change.** Each primitive should correspond to a meaningful change in robot-object interaction, object pose, gripper status, or end-effector pose.
- **Trajectory-grounded instruction.** Each instruction should correspond to a motion pattern that can be inferred from the observed trajectory.

These criteria define the atomic-action vocabulary and guide both VLM-based segmentation and human verification.

Table 5: Atomic action categories organized by interaction context and motion type.

	Object manipulation	Contact interaction	Free-space motion
Gripper	grasp / place	–	–
Translation	lift / lower / transport	push / pull / press	move
Rotation	reorient	–	rotate
Stationary		wait	

A.3 Primitive Definitions

The specific definitions of the action primitives are provided in Table 6, and visualizations of more atomic actions are provided in Figure 4. Further details are elaborated below.

Definition convention. We characterize each primitive by its dominant physical effect within an atomic segment, using end-effector motion, gripper state, object status, and contact mode as cues. For directional translations, we adopt a robot base-frame convention: $+x$ denotes forward, $-x$ backward, $+y$ left, $-y$ right, and $+z$ upward, while $-z$ denotes downward. Under this convention, primitives such as *transport* and *move* are annotated with respect to base-frame directions.

Composite primitive representation. Real robot demonstrations may contain overlapping motion patterns. Most segments are represented by a single primitive, while some segments contain multiple inseparable action components. The corresponding natural-language instruction describes the retained components in a vision-agnostic form, together with the object-holding state. In the action-category statistics, a composite episode is counted once under each primitive it contains.

Rotational primitives. Rotational actions are represented as either target-oriented reorientation of a held object or free-space rotation of the empty gripper. Orientation-changing actions applied to a held object are denoted as *reorient*, e.g., upright, flat, vertical, or horizontal, while free-space rotations of the empty gripper are denoted as *rotate*.

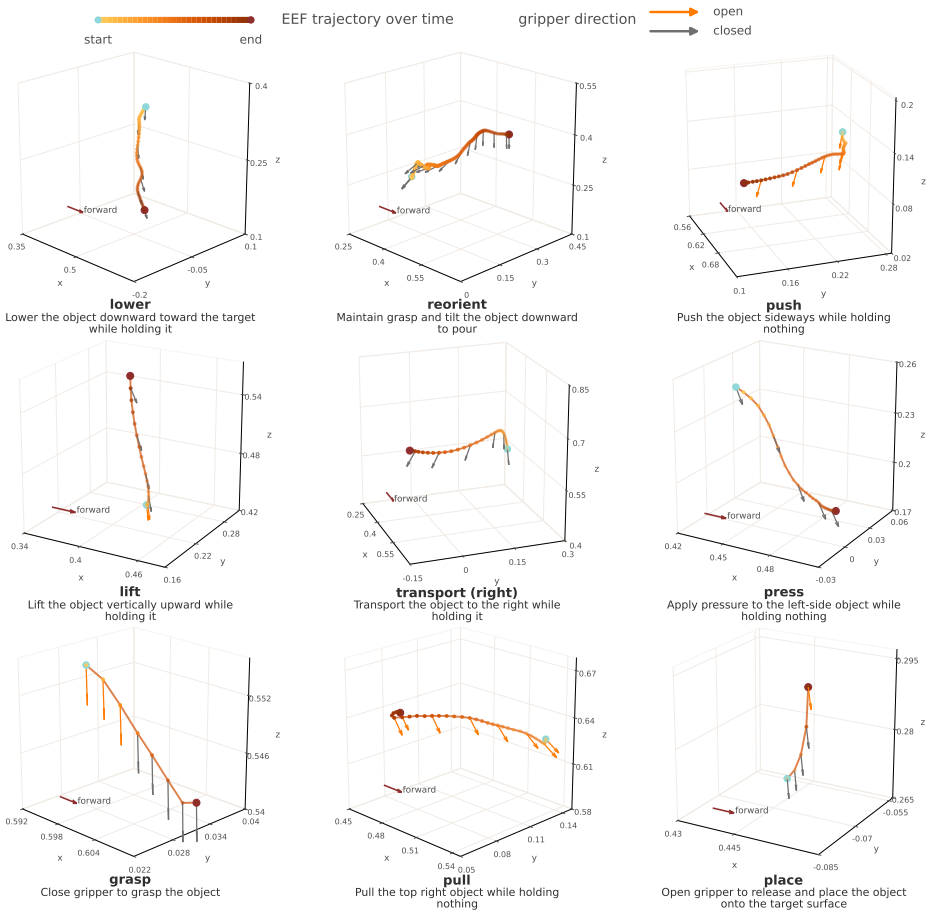


Figure 4: Additional examples of action primitives.

Table 6: Primitive definitions and typical motion patterns in the LA dataset.

Primitive	Definition and Typical Motion Pattern
move	Move the empty gripper in free space using base-frame directions.
grasp	Align with a target object and close the gripper to establish a grasp.
lift	Move a held object primarily along $+z$.
transport	Move a held object in the horizontal plane along base-frame directions.
place	Move a held object to a target pose or support region and open the gripper.
reorient	Adjust a held object's orientation toward a target state (e.g., upright, flat).
lower	Move a held object primarily along $-z$.
wait	Maintain the current end-effector pose and gripper state over a short stationary interval.
push	Apply contact-based motion to move an object away from the end effector.
press	Apply downward contact force to activate or depress a target.
pull	Apply contact-based motion to move an object toward the end effector.
rotate	Rotate the empty gripper in free space around its axis.

A.4 Prompt Construction Details

Prompt components. The prompt guides the VLM to identify short atomic segments with structured subaction strings and vision-agnostic instructions. It includes the original task description, sampled multi-view video frames, keyframe events, the atomic-action vocabulary, primitive definitions, category assignment rules, and the required JSON output schema. The prompt also instructs the VLM to jointly reason over all available views.

Design goals. The prompt serves three roles in dataset construction. First, it reduces label drift by constraining outputs to a predefined atomic-action taxonomy. Second, it demonstrates the division of atomic actions at different granularities, including relatively short gripper actions, relatively long transport or reorient actions, and composite actions. Third, it requires both a structured action label and a natural-language instruction. The instruction is written in a vision-agnostic form, describing action type, direction, and optional object or object-status information, while avoiding object identity, appearance, material, and environmental details. This format supports LA pretraining where visual input is masked.

Keyframe constraints. Keyframes provide temporal cues for identifying atomic-action boundaries. Gripper open/close events indicate likely manipulation transitions, while static keyframes often correspond to alignment, grasping, placing, or phase transitions. These cues are used as soft hints together with visual observations when determining segment boundaries.

Prompt template. The following abbreviated template summarizes the segmentation input structure:

Task context: original long-horizon task description and sampled multi-view video frames.
Action vocabulary: move, grasp, lift, transport, place, reorient, lower, wait, push, press, pull, and rotate. **Object-status tags:** append `|HOLDING` or `|EMPTY` to the `subaction` string, depending on whether an object is held by the gripper. **Primitive definitions:** specific definitions and descriptions of different action primitives **Keyframe cues:** timestamped static, gripper-open, and gripper-close events. **Instruction rules:** describe action type, direction, and optional object-status information while avoiding object identity and appearance details. **Output fields:** `subaction`, `instruction`, `start`, and `end` in JSON format. **Granularity examples:** decomposition examples such as decompose place cup into move $\rightarrow$ grasp $\rightarrow$ lift $\rightarrow$ transport $\rightarrow$ place.

A.5 VLM-BASED TEMPORAL SEGMENTATION

Model and inputs. We use Qwen-3-VL-Plus to propose temporal segments for each long-horizon demonstration. Video frames are sampled at 2 FPS and resized to 224×224. When multiple camera views are available, each view is provided as a separate video stream with an associated textual view label. The VLM is provided with the original task description, the segmentation prompt, and formatted keyframe events as additional temporal cues.

Output format. The VLM outputs a JSON list of candidate atomic-action segments. Each segment contains a structured `subaction` string, a vision-agnostic instruction, and start/end timestamps:

```
[
  {
    "subaction":  "lift|HOLDING",
    "instruction":  "lift the object upward while holding
it",
    "start":  3.50,
    "end":  5.00
  }
]
```

We use structured JSON outputs with a sampling temperature of 0.2 to improve consistency. Failed or malformed outputs are automatically retried.

Proposal generation. The VLM is used as a proposal generator for candidate temporal segmentation. It integrates visual observations, task context, keyframe cues, and taxonomy constraints to produce initial segment proposals. These proposals are subsequently filtered and refined through human verification and quality control before being included in the final LA dataset.

Cost Analysis. Segmenting a subset of the DROID dataset comprising 9,560 episodes with Qwen-3-VL-Plus incurred a token consumption of 44.29M input tokens and 3.385M output tokens. On

average, each episode required 4.633K input tokens and 0.354K output tokens. These statistics quantify the token usage of VLM-based dataset segmentation.

A.6 Human Verification Protocol

Annotation target. Human verification is applied to the VLM-generated candidate atomic-action segments before they are included in the LA dataset. For each candidate, annotators inspect the corresponding video segment together with the structured subaction label and the generated instruction. They evaluate whether the temporal boundary, action label, and instruction are mutually consistent. This step filters noisy VLM proposals, including over-segmented clips, merged primitives, imprecise boundaries, and instructions that are not well aligned with the observed action.

Quality score. Each segment receives a quality score from 0 to 3, as shown in Table 7. The score measures whether the segment is suitable for LA dataset training in terms of temporal boundary quality, action-label correctness, and language-action alignment.

Table 7: Quality score used for human verification.

Score	Meaning	Interpretation
0	Unusable	The segment is severely misaligned or does not contain a meaningful atomic action.
1	Low quality	The segment contains relevant motion but the boundary, label, or instruction is too noisy for reliable training.
2	Acceptable	The segment is mostly correct and useful for training with minor imperfections.
3	High quality	The segment, subaction label, and instruction are well aligned.

Confidence score. Annotators also provide a confidence score from 0 to 2, as shown in Table 8. This score records annotator certainty in the quality judgment and is used to monitor ambiguous cases and annotation reliability. It is not used in the final filtering stage. The average confidence score is 1.97, indicating generally high annotator certainty.

Table 8: Confidence score recorded during human verification.

Score	Level	Interpretation
0	Low	The segment is ambiguous or difficult to assess.
1	Medium	The annotator can assess the segment but some uncertainty remains.
2	High	The annotator is confident in the quality judgment.

Agreement evaluation. To evaluate annotation consistency, 10% of the generated segments are annotated by at least two annotators. We measure inter-annotator agreement using Krippendorff's alpha on an interval scale, treating the quality scores from 0 to 3 as equally spaced, and obtain $\alpha = 0.7794$. This result suggests a reasonably consistent level of agreement across annotators for the verification task.

Filtering rule. The cleaned LA dataset retains segments with quality score at least 2 and discards segments with quality score below 2. For samples annotated by multiple annotators, we use the average quality score for filtering. Thus, retained segments must be rated at least acceptable in terms of temporal boundary quality, action-label correctness, and language-action alignment. This rule removes low-quality VLM proposals while preserving segments suitable for LA dataset training. VLM-based segmentation produces 5.95 candidate atomic episodes per original VLA episode on average, and 58.2% of these candidates pass the quality threshold.

A.7 Dataset Statistics

Applying the construction pipeline to DROID (Khazatsky et al., 2024) yields LA-33K. Table 9 summarizes the scale and episode granularity across the processing phases.

Table 9: Dataset scale and episode granularity during LA dataset construction.

Processing Phase	Episodes	Frames	Frames / Ep.
Original VLA episodes	9,560	2,751,615	287.83
VLM-generated LA candidates	56,899	2,551,102	44.84
Final human-verified LA episodes	33,116	1,524,990	46.05

Table 10: LA4VLA-1B model configuration.

Component	Configuration
Backbone	InternVL3-1B
Action head	Flow-matching transformer
Action-head token dimension	896
Action-head transformer layers	8
Action-head attention heads	8
Action-head MLP hidden dimension	1024
Flow-matching inference steps	50

B Model and Training Details

B.1 Model Architecture and Policy Interface

Table 10 summarizes the implementation-specific architecture settings used for LA pretraining and simulated downstream experiments.

B.2 Data Representation and Visual Masking

For LA pretraining and simulated downstream experiments, each sample contains a language instruction, a proprioceptive state vector, and a target action chunk. LA-33K removes visual observations, whereas LA-33K-V uses the corresponding original RGB observations. When present, images are resized to 448×448. State and per-step action vectors are padded to 24 dimensions and normalized using statistics computed from the corresponding training data. The action horizon is 50. For both LA-33K and LA-33K-V, actions are represented as relative end-effector commands together with the gripper command. Thus, LA training changes the available conditioning signal rather than the prediction target. For MixPT, visual input is disabled for LA-33K samples and enabled for LA-33K-V samples.

B.3 Optimization and Finetuning

Table 11 summarizes the shared two-stage optimization schedule used by all pretraining paradigms. Stage 1 trains the action head for 600 steps, and Stage 2 trains the language model together with the action head for a total of 11,000 steps. Stage 2 initializes from the final Stage 1 checkpoint while resetting the optimizer and learning-rate scheduler. We use AdamW with linear learning-rate warmup followed by cosine decay. The paradigms differ only in the data format and ordering used within these stages, as summarized in Table 12. Each paradigm therefore uses the same total budget of 11,600 optimization steps.

Here LA denotes LA-33K and VLA denotes LA-33K-V. For LA-VLA, the optimizer and learning-rate scheduler are reset when switching from the LA phase to the VLA phase. Apart from the data schedules and associated visual conditioning, the model architecture, trainable modules, batch sizes, and base optimization hyperparameters are shared across paradigms.

Table 13 summarizes the shared downstream finetuning configuration used for MetaWorld, LIBERO, and LIBERO-Plus. Within each benchmark, the compared variants differ only in the model weights used to initialize finetuning.

Table 11: Shared two-stage optimization configuration for pretraining.

Setting	Stage 1	Stage 2
Trainable modules	Action head	Language model + action head
Initialization	InternVL3-1B backbone	Stage 1 checkpoint at step 600
Training steps	600	11,000
Batch size	512	384
Peak learning rate	3.0×10^{-5}	2.5×10^{-5}
Warmup steps	28	36
Weight decay	1.0×10^{-4}	1.0×10^{-4}
Dropout	0.2	0.2
Gradient clipping norm	1.0	1.0

Table 12: Pretraining data schedules.

Paradigm	Stage 1 (600 steps)	Stage 2 (11,000 steps)	Visual input
LA	LA	LA	Masked
VLA	VLA	VLA	Enabled
LA-VLA	LA	5.5k LA $\rightarrow$ 5.5k VLA	Masked $\rightarrow$ enabled
MixPT	LA/VLA (1:1)	LA/VLA (1:1)	Source-dependent

C Simulation Evaluation Details

C.1 Simulation Benchmarks

MetaWorld. MetaWorld (Yu et al., 2020) is designed to assess whether a policy can reuse shared control structure across diverse manipulation skills. Its 50 tasks share a common tabletop environment and robot embodiment but require distinct behaviors, ranging from reaching and pushing to articulated-object manipulation. We use MetaWorld to evaluate broad multi-task manipulation competence and transfer across distinct task families, and report results over the Easy, Medium, Hard, and Very Hard task groups. Representative task executions are shown in Figure 5.

LIBERO. LIBERO (Liu et al., 2023) is designed to disentangle knowledge transfer in lifelong robot learning. LIBERO-Spatial changes spatial relations among otherwise identical objects, testing whether a policy can distinguish and retain spatial knowledge. LIBERO-Object changes object identities under shared layouts, testing object-level declarative knowledge transfer. LIBERO-Goal fixes the objects and their spatial configuration while changing the task goal, testing transfer of procedural knowledge about motions and behaviors. LIBERO-Long contains long-horizon manipulation tasks that require composing multiple skills. Together, these four suites evaluate whether learned representations transfer across spatial, object, goal, and temporal variations. Representative task executions are shown in Figure 6.

LIBERO-Plus. LIBERO-Plus (Fei et al., 2025) is a controlled robustness diagnostic for VLA models. Rather than introducing new manipulation skills, it perturbs one factor at a time during evaluation to isolate sources of policy brittleness. In this work, we evaluate robustness to object-layout changes, background-texture changes, sensor noise, and camera-viewpoint shifts, which respectively probe robustness to object configuration, scene appearance, image corruption, and viewpoint variation. Representative executions under these perturbations are shown in Figure 7.

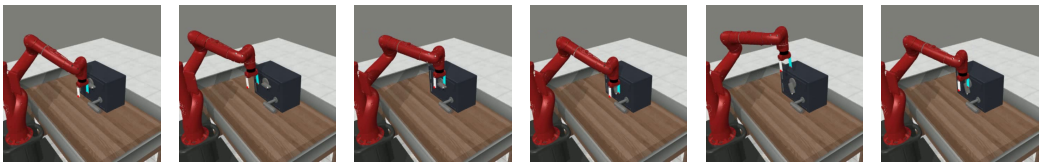
C.2 Evaluation Protocol

For all simulation benchmarks, a rollout is counted as successful when the benchmark-provided task-completion signal is triggered before episode termination.

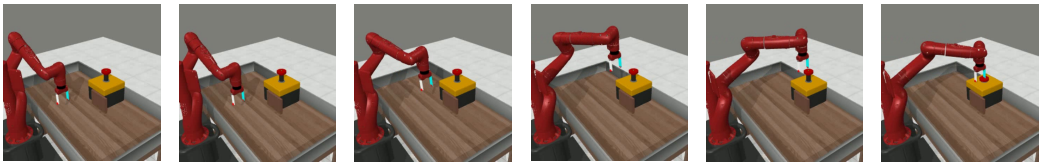
MetaWorld. Following the evaluation protocol of Evo-1 (Lin et al., 2026b), we evaluate the 28 Easy, 11 Medium, 6 Hard, and 5 Very Hard MetaWorld tasks over 10 rollouts per task, for a total of 500 rollouts per policy. We first compute the success rate within each task group and report their equally weighted average.

Table 13: Shared simulation finetuning configuration.

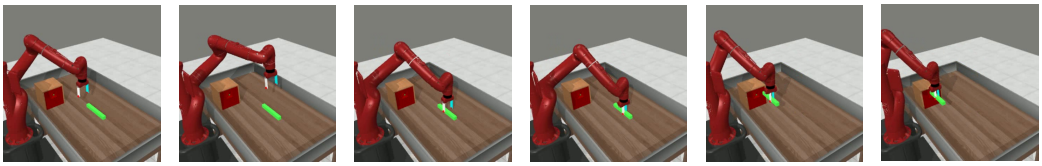
Setting	Configuration
Benchmarks	MetaWorld, LIBERO, and LIBERO-Plus
Visual conditioning	Enabled
Trainable modules	VLM + action head
Training steps	20,000
Batch size	192
Peak learning rate	6.0×10^{-5}
Warmup steps	200
Weight decay	5.0×10^{-4}
Dropout	0.2
Gradient clipping norm	1.0
Image augmentation	Enabled



Door Lock: "Lock the door by rotating the lock clockwise"



Button Press: "Bypass a wall and press a button from the top"



Peg Insertion: "Insert a peg sideways"

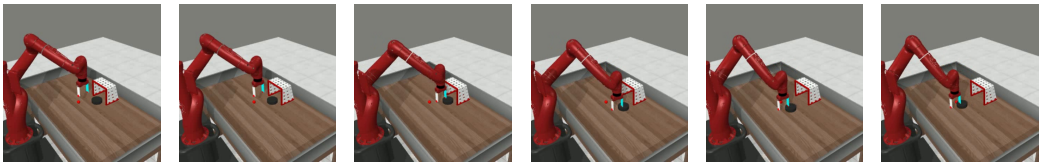


Plate Slide: "Get a plate from the cabinet"

Figure 5: Representative task executions in MetaWorld. Each row shows temporally ordered frames from one task rollout.

LIBERO. We evaluate each task over 10 rollouts under each of five evaluation seeds, yielding 50 rollouts per task and 2,000 rollouts per policy across the 40 tasks. We first compute the success rate for each of the LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long suites, and report the equally weighted average across the four suites.

LIBERO-Plus. We use the official LIBERO-Plus evaluation instances for four selected perturbation categories. The Objects Layout, Background Textures, Sensor Noise, and Camera Viewpoints conditions contain 312, 289, 449, and 419 evaluation rollouts, respectively. We compute the success rate for each condition and report the equally weighted average across the four perturbation categories.

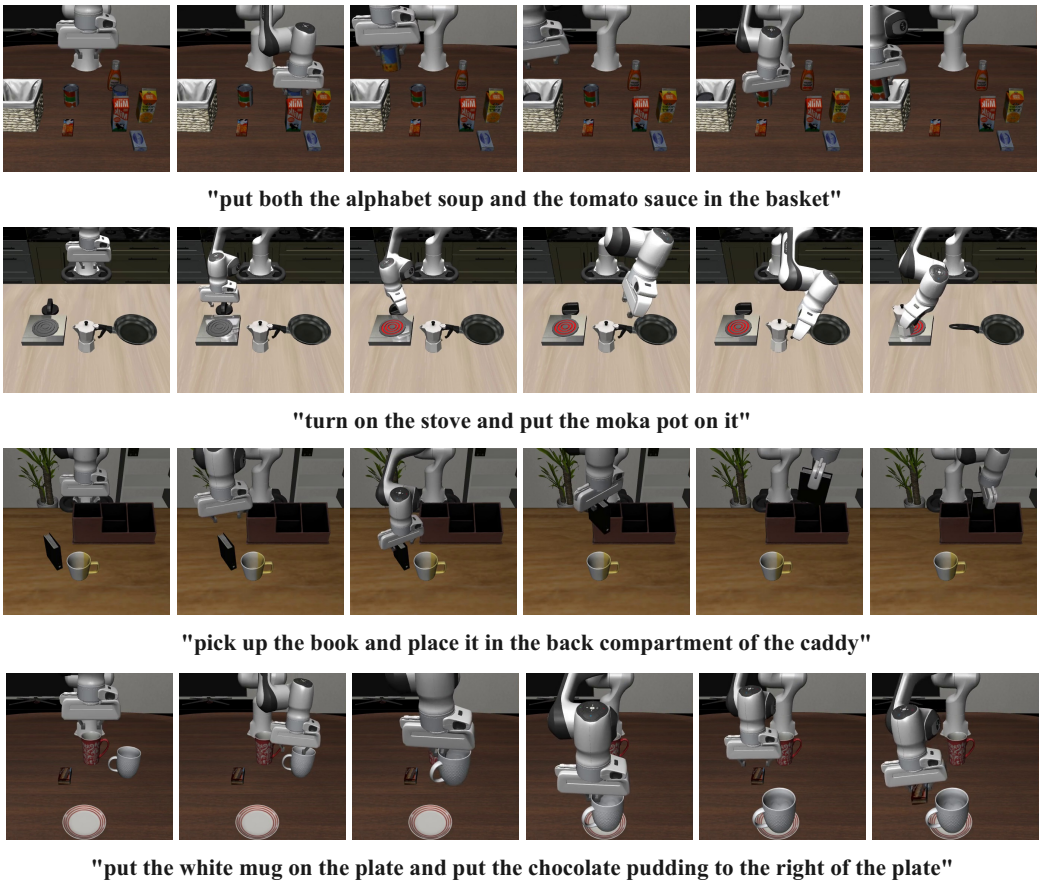


Figure 6: Representative task executions in LIBERO. Each row shows temporally ordered frames from one task rollout.

C.3 Comparison with Conventional VLA Pretraining

The main experiments compare LA-33K with its paired VLA counterpart LA-33K-V, both constructed from the same atomic action segments. To additionally compare with conventional VLA pretraining, we pretrain LA4VLA-1B directly on the same 9,560 original DROID demonstrations without atomic segmentation, retaining their original task instructions and visual observations. We denote this baseline as *DROID-VLA* and refer to pretraining on LA-33K-V as *Atomic VLA*. All variants use the same model architecture, pretraining steps and hyperparameters, and downstream finetuning protocol.

Table 14: Comparison with conventional VLA pretraining on unsegmented DROID demonstrations. Success rates are reported in percent.

MetaWorld					
Pretraining	Easy	Medium	Hard	Very Hard	Avg.
Atomic VLA	88.6	68.2	68.3	**94.0**	79.78
DROID-VLA	88.2	74.5	**71.7**	86.0	80.11
LA	**93.2**	**81.8**	65.0	92.0	**83.00**
LIBERO-Plus					
Pretraining	Layout	Background	Noise	Camera	Avg.
Atomic VLA	71.25	79.52	68.24	17.02	59.01
DROID-VLA	78.04	83.22	65.12	14.99	60.34
LA	**81.41**	**84.08**	**78.62**	**31.98**	**69.02**

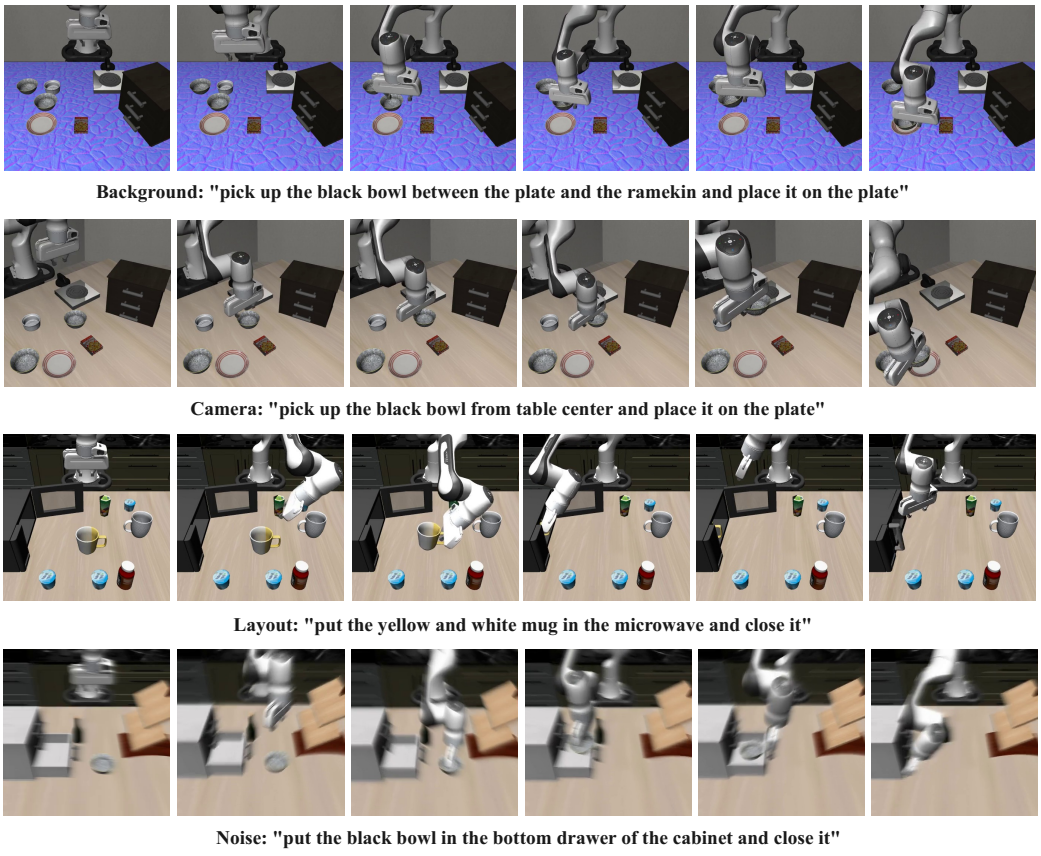


Figure 7: Representative task executions under the evaluated LIBERO-Plus perturbations. Each row shows temporally ordered frames from one task rollout.

DROID-VLA reaches an average success rate of 80.11% on MetaWorld, slightly above Atomic VLA at 79.78% but below LA at 83.00%. On LIBERO-Plus, DROID-VLA similarly improves over Atomic VLA from 59.01% to 60.34%, while LA reaches 69.02%. Thus, LA pretraining outperforms conventional unsegmented VLA pretraining by 2.89 and 8.68 percentage points on MetaWorld and LIBERO-Plus, respectively. Unlike the paired LA–Atomic VLA comparison, this experiment also changes trajectory and instruction granularity, and therefore serves as a practical comparison with conventional VLA pretraining rather than a factor-isolation ablation.

D Real-World Experimental Setup

Platform. The real-world experiments use a UFactory xArm6 arm with a parallel-jaw gripper. The policy receives two RGB camera views, one from a wrist-mounted camera and one from a third-person camera.

Tasks. We design three language-conditioned manipulation tasks. In *Press Button*, four buttons are placed at different positions on the table, and the policy must press the button specified by the instruction. In *Place Book*, the policy must insert a book into one of three shelf positions specified by the instruction. In *Place Drink*, the policy must place a drink into the cell of a 3×3 grid specified by the instruction. Within each task, the initial scene and observation are shared across target goals, so the visual input cannot disambiguate the target and the instruction alone determines which goal to execute.

Demonstrations and training. We collect 100 teleoperated demonstrations per task, 300 in total. All four policy variants are finetuned on the same demonstrations under the same protocol, and each uses a single multi-task checkpoint evaluated on all three tasks.

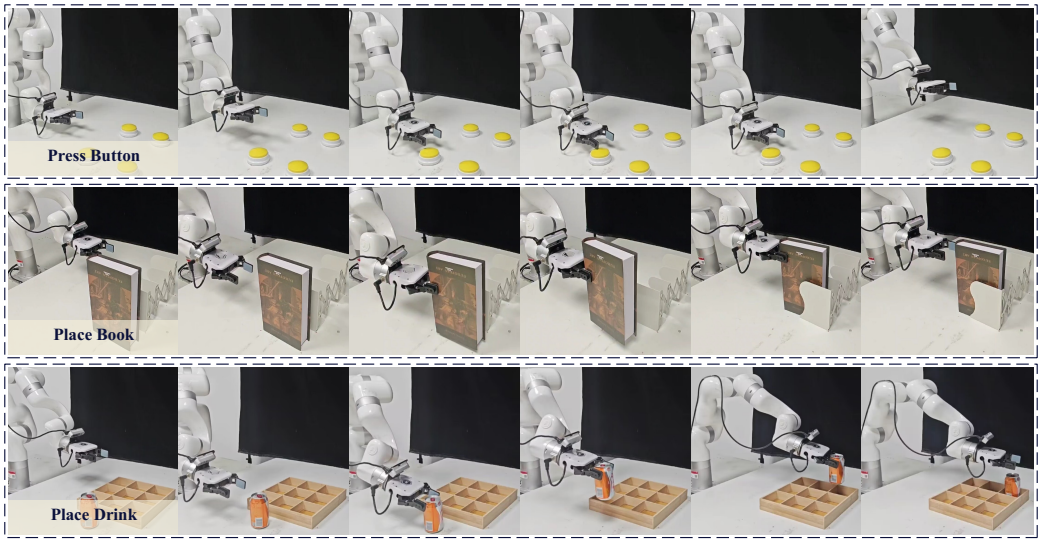


Figure 8: Real-world task execution on the xArm6 platform. Each row shows a representative rollout of one language-conditioned task, with frames ordered from left to right over the course of execution: *Press Button* (top), *Place Book* (middle), and *Place Drink* (bottom).

Evaluation protocol. In *Press Button*, the four target buttons are evaluated in a fixed cyclic order: top-right, bottom-right, bottom-left, and top-left, repeated for 5 rounds, giving 20 trials in total. In *Place Book*, the three shelf positions are tested 7, 6, and 7 times, giving 20 trials. In *Place Drink*, four target cells are each evaluated 5 times, giving 20 trials. Each trial is capped by a step limit of 800 low-level control steps for Press Button, 1,200 for Place Book, and 1,000 for Place Drink.

Success criteria. Press Button counts as a success if the commanded button is pressed at least once within the step limit, and as a failure otherwise. Place Book counts as a success if the book is inserted to at least half of the shelf depth. Place Drink counts as a success if the drink is placed into the commanded cell in an upright and stable configuration without tipping over within the step limit. Prematurely opening the gripper, causing the drink to spill, is also considered a failure.

Gaussian image noise. For the noisy-observation evaluation, we apply independent additive zero-mean Gaussian noise to every pixel and channel of both camera observations after spatial preprocessing to 448×448. We use a standard deviation of $\sigma = 5$ in the 8-bit pixel domain ($5/255$ on the normalized scale) for Press Button and $\sigma = 15$ ($15/255$) for Place Drink. The resulting pixel values are rounded and clipped to $[0, 255]$. The proprioceptive state and language instruction remain unchanged. Each task is evaluated over 20 trials under the noisy-observation condition.

E Metrics for Instruction Following Analysis

Notations. The evaluation set is denoted as

$$\mathcal{D} = \{(y_i, \hat{\mathbf{p}}_i, \mathbf{g}_i, \mathbf{u}_i)\}_{i=1}^{N},$$

where $y_i \in \mathcal{Y}$ is the instruction direction label, $\hat{\mathbf{p}}_i \in \mathbb{R}^3$ is the predicted endpoint displacement, $\mathbf{g}_i \in \mathbb{R}^3$ is the ground-truth endpoint displacement under the corresponding instruction, and $\mathbf{u}_i \in \mathbb{R}^3$ is the unit vector corresponding to the instruction-specified canonical direction. For example, $\mathbf{u}_i = (1, 0, 0)$ for x^+ and $\mathbf{u}_i = (-1, 0, 0)$ for x^-. The endpoint displacements are obtained by summing translational increments over the prediction horizon:

$$\hat{\mathbf{p}}_i = \sum_{t=1}^{T} \Delta\hat{\mathbf{x}}_{i,t}, \qquad \mathbf{g}_i = \sum_{t=1}^{T} \Delta\mathbf{x}_{i,t}^{\mathrm{GT}}.$$

Directional Alignment Rate (DAR). DAR measures the fraction of trials whose predicted endpoint displacement lies in the half-space aligned with the canonical instruction direction:

$$\mathrm{DAR} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\left[\hat{\mathbf{p}}_i^\top \mathbf{u}_i > 0\right].$$

Direction Consistency Score (DCS). DCS measures the cosine similarity between the predicted endpoint displacement and the ground-truth endpoint displacement:

$$\mathrm{DCS} = \frac{1}{N}\sum_{i=1}^{N} \frac{\hat{\mathbf{p}}_i^\top \mathbf{g}_i}{\|\hat{\mathbf{p}}_i\|_2 \, \|\mathbf{g}_i\|_2}.$$

Since $\mathbf{g}_i$ is collected under the corresponding instruction, DCS evaluates whether the predicted motion follows the instruction-conditioned demonstrated direction. In implementation, trials with near-zero predicted or ground-truth displacement norm are assigned a cosine value of 0.

Separability Ratio (SR). SR compares two quantities: the average distance between predictions with different instruction labels, and the average distance between predictions with the same instruction label. For each instruction label $k \in \mathcal{Y}$, let

$$\mathcal{I}_k = \{i : y_i = k\}$$

be the indices of trials labeled as direction k.

First compute the average within-direction distance by averaging pairwise distances within each direction label and then averaging over direction labels:

$$d_{\mathrm{intra}} = \frac{1}{|\mathcal{Y}|}\sum_{k\in\mathcal{Y}} \frac{2}{|\mathcal{I}_k|(|\mathcal{I}_k|-1)} \sum_{\substack{i,j\in\mathcal{I}_k \\ i<j}} \|\hat{\mathbf{p}}_i - \hat{\mathbf{p}}_j\|_2.$$

Then compute the average between-direction distance by averaging pairwise distances for each direction pair and then averaging over direction pairs:

$$d_{\mathrm{inter}} = \frac{2}{|\mathcal{Y}|(|\mathcal{Y}|-1)} \sum_{\substack{a,b\in\mathcal{Y} \\ a<b}} \frac{1}{|\mathcal{I}_a||\mathcal{I}_b|} \sum_{i\in\mathcal{I}_a}\sum_{j\in\mathcal{I}_b} \|\hat{\mathbf{p}}_i - \hat{\mathbf{p}}_j\|_2.$$

Here $a < b$ simply means that each pair of different direction labels is counted once. The separability ratio is then defined as follows, and here ϵ denotes a small positive constant used for numerical stability:

$$\mathrm{SR} = \frac{d_{\mathrm{inter}}}{d_{\mathrm{intra}} + \epsilon}.$$

A higher ratio means that the predicted endpoint displacements are more separable by instruction direction.

Silhouette Score (SS). SS measures how well endpoint displacements cluster according to instruction direction. For each trial i, define the average distance to trials with the same instruction direction as

$$a_i = \frac{1}{|\mathcal{I}_{y_i}| - 1} \sum_{\substack{j\in\mathcal{I}_{y_i} \\ j\neq i}} \|\hat{\mathbf{p}}_i - \hat{\mathbf{p}}_j\|_2,$$

and the nearest average distance to another instruction-direction cluster as

$$b_i = \min_{k\neq y_i} \frac{1}{|\mathcal{I}_k|} \sum_{j\in\mathcal{I}_k} \|\hat{\mathbf{p}}_i - \hat{\mathbf{p}}_j\|_2.$$

The silhouette value for trial i is

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i) + \epsilon},$$

and the overall silhouette score is

$$\mathrm{SS} = \frac{1}{N}\sum_{i=1}^{N} s_i.$$

Higher SS indicates that endpoint displacements are closer to trials from the same instruction direction than to trials from other directions.

F Instruction-Following Behavior and Representation Analysis

This appendix provides the full evaluation settings, quantitative results, trajectory visualizations, and t-SNE figures for the instruction-following behavior and representation analysis summarized in the main text (Section 5.6).

Evaluation settings. The LA-pretrained policy predicts actions from the language instruction and robot state, with visual input masked. We evaluate two settings adapted from Section 3. In the *LA-pretrained* condition, the policy receives the instruction and corresponding robot state from a separate evaluation set not used during pretraining. In the *Conflicting-state* condition, we keep the same evaluation instruction but replace its robot state with one from an LA pretraining pair whose action trajectory moves in the opposite direction. Since visual input is masked, the robot state is the only non-language input that can conflict with the instruction.

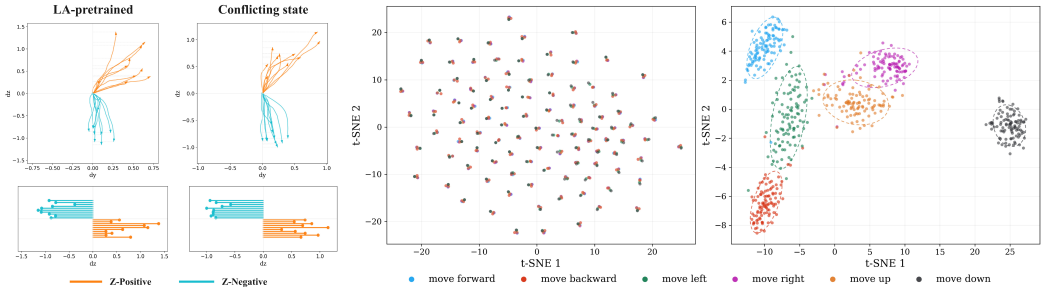


Figure 9: Direction-following behavior and representation analysis of the LA-pretrained policy. **Left panel:** qualitative direction-following under the *LA-pretrained* and *Conflicting-state* conditions for a representative pair of opposite commands along the z axis. The top row shows predicted trajectories in the dy–dz plane, and the bottom row shows endpoint projections along the tested z axis. **Right panel:** t-SNE visualization of the policy representations before action decoding, colored by direction command. The VLA-trained policy maps different commands to mixed clusters, whereas the LA-pretrained policy forms clearer command-aligned clusters.

Table 15: Quantitative direction-following results of the LA-pretrained policy under masked visual input. Metrics are computed over 100 direction-following cases for each evaluation condition.

Condition	DAR ↑	DCS ↑	SR ↑	SS ↑
LA-pretrained	**0.99**	**0.85**	**1.46**	**0.20**
Conflicting state	0.95	0.80	1.41	0.16

Direction-following results. The left panels of Figure 9 visualize the predicted trajectories and endpoint projections under the *LA-pretrained* and *Conflicting-state* conditions for the same representative pair of opposite commands as in Section 3. Table 15 reports the corresponding directional metrics. Under the *LA-pretrained* condition, the policy predicts action directions that closely follow the instruction, achieving a DCS of 0.85 and a DAR of 0.99. The trajectories also show clear endpoint separation between opposite instructions, indicating reliable instruction-conditioned direction following without visual input. The *Conflicting-state* condition further tests whether this behavior remains when the input robot state is replaced with the state from an instruction-action pair whose motion direction is opposite to the current instruction. In this setting, DCS remains high at 0.80, close to the *LA-pretrained* value of 0.85. This indicates that the predicted action direction follows the instruction rather than being dominated by the substituted robot state. Overall, these results show that LA pretraining yields more reliable instruction-conditioned direction following, even under the conflicting-state condition.

Representation analysis. We next examine whether the direction-following results are reflected in the internal representations used for action prediction. For both policies, we extract the policy representations fed into the action head and project them with t-SNE. The representations are collected under masked visual input with the same robot states and direction instructions along the three Cartesian axes. The resulting t-SNE visualizations are shown in the right panels of Figure 9.

The two policies organize this representation space differently. For the VLA-trained policy, the projected representations form many small clusters, each containing multiple command directions. This suggests that the representations are primarily grouped by robot state, while the instruction has only a weak effect on the representation used for action decoding. This is consistent with its weak direction following under the Visual-removed-input condition in Table 1. In contrast, the LA-pretrained policy forms clearer clusters aligned with the commanded directions, and opposite commands fall into distinct regions. This suggests that LA pretraining induces a more instruction-conditioned action representation, consistent with the direction-following results above.

Taken together, the direction-following and representation results indicate that LA pretraining produces a policy whose predicted actions are more strongly conditioned on the language instruction. The policy predicts instruction-aligned directions without visual input, maintains this behavior when the input robot state is replaced with state information from an opposite-direction instruction-action pair, and forms action-decoding representations that are more clearly separated according to the instructed direction. These properties are obtained before downstream finetuning and provide behavioral and representational support for the downstream gains of policies with LA pretraining.