

LIBERO-Para: A Diagnostic Benchmark and Metrics for Paraphrase Robustness in VLA Models

Chanyoung Kim^{1*}, Minwoo Kim^{2*}, Minseok Kang², Hyunwoo Kim², Dahuin Jung^{2†}

¹School of Computer Science and Engineering, Soongsil University, Seoul, Korea

²Department of Artificial Intelligence, Chung-Ang University, Seoul, Korea
dahuinjung@cau.ac.kr

Abstract

Vision–Language–Action (VLA) models achieve strong performance in robotic manipulation by leveraging pre-trained vision–language backbones. However, in downstream robotic settings, they are typically fine-tuned with limited data, leading to overfitting to specific instruction formulations and leaving robustness to paraphrased instructions underexplored. To study this gap, we introduce LIBERO-Para, a controlled benchmark that independently varies action expressions and object references for fine-grained analysis of linguistic generalization. Across seven VLA configurations (0.6B–7.5B), we observe consistent performance degradation of 22–52 pp under paraphrasing. This degradation is primarily driven by object-level lexical variation: even simple synonym substitutions cause large drops, indicating reliance on surface-level matching rather than semantic grounding. Moreover, 80–96% of failures arise from planning-level trajectory divergence rather than execution errors, showing that paraphrasing disrupts task identification. Binary success rate treats all paraphrases equally, obscuring whether models perform consistently across difficulty levels or rely on easier cases. To address this, we propose PRIDE, a metric that quantifies paraphrase difficulty using semantic and syntactic factors. Our benchmark and code are available at <https://github.com/cau-hai-lab/LIBERO-Para>.

1 Introduction

Vision–Language–Action (VLA) models have emerged as a promising paradigm for robotic manipulation (Zitkovich et al., 2023; Black et al., 2024). By leveraging large pre-trained vision–language models (VLMs) as backbones, VLA models acquire instruction-following capabilities through large-scale multimodal perception (Zitkovich et al., 2023; Black et al., 2024). To

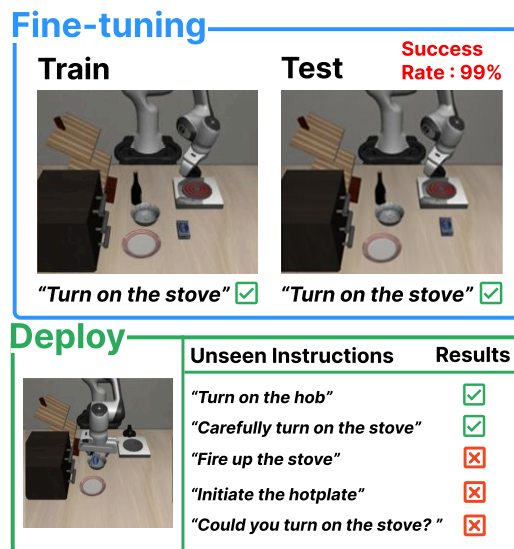


Figure 1: Illustration of paraphrase robustness gap under data-scarce fine-tuning: VLA models can overfit to seen instruction phrasings during fine-tuning and fail to generalize to paraphrased variants at deployment.

deploy such models in specific environments (e.g., kitchens, homes, offices, or laundry rooms) (Black et al., 2024; Physical Intelligence et al., 2025; Zitkovich et al., 2023; Fu et al., 2024; Wu et al., 2023), existing approaches typically perform fine-tuning using environment-specific demonstration data (Fu et al., 2024). However, acquiring such data entails considerable cost and labor overhead. Consequently, real-world deployment often necessitates data-scarce fine-tuning, which can induce overfitting and degrade the general knowledge embedded in pre-trained VLA models (Yadav et al., 2025; Zhou et al., 2025). Such overfitting raises a practical concern that models may perform well on seen instruction phrasings yet fail to generalize to unseen paraphrased instructions at deployment. Under these circumstances, as shown in Fig. 1, benchmarks that rigorously assess robustness after fine-tuning become important (Zhou et al., 2025).

However, the LIBERO benchmark (Liu et al.,

*Equal contribution. †Corresponding author.

2023), which has become widely adopted in current VLA research, evaluates models under identical instructions during both training and evaluation. It primarily measures visual generalization to novel object configurations or scene layouts, while leaving robustness to linguistic variation largely unexamined. Consequently, the linguistic robustness of VLA models remains insufficiently validated (Fei et al., 2025; Wang et al., 2024b; Zhou et al., 2025).

Several benchmarks have examined linguistic variation in VLA evaluation (Mees et al., 2021; Wang et al., 2024b). However, as summarized in Tab. 1, these approaches exhibit key limitations for assessing paraphrase robustness. Paraphrasing is often treated as one axis among broader multimodal perturbations (Zhou et al., 2025; Fei et al., 2025; Wang et al., 2026), or conflated with task-level semantic changes that alter the intended behavior (Hou and Zhao, 2026), rather than isolating meaning-preserving variation. Furthermore, linguistic properties specific to robotic manipulation instructions are not explicitly modeled, and the distance between paraphrases is not formally quantified, limiting the ability to analyze which types of variation most severely impact performance.

To address these limitations, we introduce LIBERO-Para, a controlled benchmark for evaluating paraphrase robustness in VLA models, along with PRIDE (Paraphrase Robustness Index in Robotic Instructional DEviation), a metric that combines keyword similarity (lexical shift) and structural similarity (syntactic variation) with task success for fine-grained robustness analysis. LIBERO-Para is grounded in the linguistic structure of robotic manipulation instructions—where actions and objects serve as core semantic elements—and adopts a two-axis design that independently varies action expressions and object references. Our analysis reveals three key findings:

- **Paraphrase Fragility Persists:** Performance consistently degrades under paraphrased instructions across architectures, scales, and fine-tuning strategies.
- **Object-Level Bottleneck:** Object-level lexical variation is the dominant source of degradation, indicating reliance on surface-level matching rather than semantic grounding.
- **Planning-Level Failures:** 80–96% of failures arise from trajectory divergence, suggesting errors in task identification rather than action execution.

Benchmark	Scope	Paraphrase Control	Variation Axis	Para. Types
CALVIN	Instruction	×	Sentence	1
LADEV	Paraphrase	×	Sentence	1
LIBERO-PRO	Multimodal	△	Sentence	2
LIBERO-Plus	Multimodal	△	Sentence	5
LIBERO-X	Multimodal	△	Sentence	5
LangGap	Task Semantics	×	4 Semantic dims	4
LIBERO-Para (Ours)	Paraphrase	✓	Action × Object	43

Table 1: Comparison with existing benchmarks for linguistic robustness. LIBERO-Para provides full paraphrase control with fine-grained action × object variation axes and 43 linguistically grounded types. ×: not supported, △: partially supported, ✓: fully supported.

This work contributes to advancing VLA systems beyond high performance toward robustness to linguistic variation and reliable task interpretation.

2 Related Work

2.1 Vision-Language-Action Models

Vision-Language-Action (VLA) models map visual and linguistic input to robot actions. Early approaches extend LLM backbones to autoregressively decode discrete action tokens (Zitkovich et al., 2023; Kim et al., 2024). Recent work has diversified along several architectural axes: parallel decoding with action chunking, which predicts all actions in a single forward pass (Kim et al., 2025); VLMs with a flow-matching action expert, a separate decoder paired with a billion-scale VLM (Black et al., 2024; Physical Intelligence et al., 2025; Cai et al., 2026); lightweight bridge-based adaptation, which routes VLM representations to a compact policy head via cross-attention (Wang et al., 2025); and soft-prompted cross-embodiment designs, which encode embodiment-specific knowledge through learnable tokens (Zheng et al., 2025). The latter two operate at 0.6–0.9B scale, unlike earlier multi-billion-parameter designs. Despite this diversity, all models require environment-specific fine-tuning with limited demonstration data. We evaluate representatives from each family to assess whether paraphrase robustness is architecture-specific or a shared vulnerability.

2.2 Benchmarks for VLA models

A range of benchmarks have been proposed to evaluate linguistic conditioning in VLA models; Tab. 1 compares their design choices. CALVIN (Mees et al., 2021) and LADEV (Wang et al., 2024b) assess generalization to rephrased instructions, but treat paraphrasing as unstructured sentence-

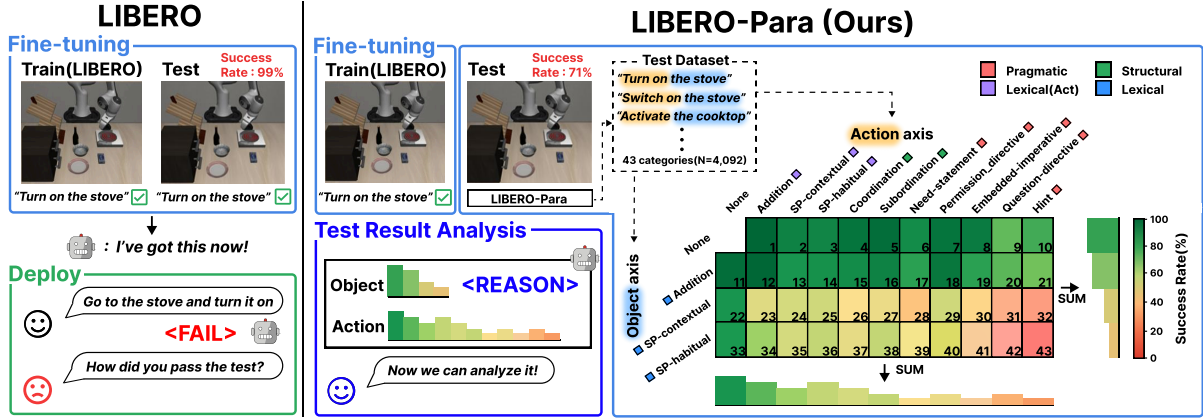


Figure 2: Overview of LIBERO-Para. Compared to LIBERO, LIBERO-Para evaluates paraphrase robustness under data-scarce fine-tuning via a controlled two-axis design (action vs. object), enabling interpretable analysis.

level variation without linguistic categorization. LIBERO-PRO, LIBERO-Plus, and LIBERO-X (Zhou et al., 2025; Fei et al., 2025; Wang et al., 2026) introduce multimodal perturbation-based evaluations that include linguistic variation as one of several axes, revealing limited dependence on genuine linguistic understanding; however, paraphrasing remains a secondary concern within their broader evaluation scope. LangGap (Hou and Zhao, 2026) targets language conditioning more directly, but its perturbations alter the intended behavior (e.g., changing which object to grasp), conflating task-level semantic changes with linguistic variation. In contrast, our LIBERO-Para differs in two key aspects. First, it isolates meaning-preserving linguistic variation from task-level semantic changes. Second, instead of sentence-level perturbations with ad-hoc categories, it identifies the essential components of manipulation instructions, action verbs and object references, and decomposes paraphrases along these axes following established linguistic taxonomies (Kovatchev et al., 2018; Ervin-Tripp, 1976). This factorization yields 43 fine-grained variation types (Tab. 1) and exposes a super-additive interaction that single-axis or sentence-level perturbation cannot capture.

3 LIBERO-Para: A Controlled VLA Benchmark for Paraphrase Robustness

We introduce LIBERO-Para, a controlled benchmark for evaluating the paraphrase robustness of VLA models. As shown in Tab. 1, existing benchmarks offer limited control over paraphrase variation; our design addresses this through a two-axis scheme that independently varies action expressions and object references—the two core linguistic

components of robotic manipulation instructions. This separation enables controlled analysis of how different linguistic factors affect VLA performance. LIBERO-Para is constructed on top of LIBERO-Goal, a setting in which linguistic understanding is essential: all tasks start from an identical initial state, making the instruction the sole cue for task identification. We generate the benchmark by paraphrasing only the instructions while keeping all other factors fixed. All paraphrases are held out for evaluation, allowing assessment of generalization to unseen linguistic variations under data-scarce fine-tuning scenarios, as illustrated in Fig. 2.

3.1 Action Variation

The action axis captures variation in how actions are linguistically expressed. We define three types of action variation grounded in established paraphrase taxonomies. (1) Lexical variation modifies the action verb at the word level, including synonym substitution and adverb insertion. (2) Structural variation alters the sentence-level realization of the action expression, such as coordination and subordination. (3) Pragmatic variation expresses actions indirectly, covering indirect speech acts. Lexical and structural variations are instantiated based on the Extended Paraphrase Typology (Kovatchev et al., 2018). Pragmatic variations are defined in accordance with Ervin-Tripp (1976). Fig. 3 (bottom) presents representative examples for each type.

3.2 Object Variation

The object axis captures variation in how objects are referenced in instructions. In robotic manipulation, object references are typically realized as noun phrases with limited complexity (e.g., “the

Original : Turn on the stove	Obj
Addition: Turn on the gas stove	
SP-contextual: Turn on the cooktop	
SP-habitual: Turn on the cooker	
	Action
Addition: Carefully turn on the stove	
SP-context: Switch on the stove	
SP-habitual: Fire up the stove	
Coordination: Go to the stove and turn it on	
Subordination: Turn on the stove so that it becomes hot	
Need-statement: I need the stove turned on	
Embedded-imperative: Could you turn on the stove?	
Permission-directive: May I have the stove turned on?	
Question-directive: Is the stove heating up?	
Hint: The stove is still off	

Figure 3: Examples of axis-specific paraphrases. Object variations modify target object references (e.g., same-polarity substitution, addition), while action variations cover lexical, structural, and pragmatic realizations grounded in established taxonomies.

stove” → “the cooktop”). We focus on lexical-level variation. Following the Extended Paraphrase Typology (Kovatchev et al., 2018), we define three subtypes: addition and same-polarity substitution (contextual and habitual variants). Fig. 3 (top) illustrates representative examples.

3.3 Compositional Variation

Beyond individual axes, we evaluate compositional paraphrases varying both action and object expressions. This setting analyzes whether the two axes have independent or interaction effects on VLA performance, which cannot be determined from single-axis perturbations by construction. Fig. 2 presents an example with a success-rate grid over variation combinations and per-axis success rates.

To ensure balanced evaluation, the benchmark includes approximately 100 samples per variation type, resulting in a total of 4,092 paraphrased instructions. Additional details on the taxonomy, paraphrase generation process, and excluded variation types—including justifications for why certain types are inapplicable to robotic manipulation instructions—are provided in Appendix A.

4 PRIDE: Paraphrase Robustness Index in Robotic Instructional DEviation

Existing VLA benchmarks rely on a binary success metric, assigning 1 if the robot completes the instructed task and 0 otherwise. However, this metric does not distinguish between easy and difficult paraphrases, obscuring whether performance re-

flects robust linguistic understanding or reliance on simpler instruction variants.

To enable interpretable robustness evaluation, we propose PRIDE, a metric that quantifies the linguistic deviation between an original instruction and its paraphrase. Unlike general-purpose metrics, PRIDE is tailored to robotic instructions and decomposes paraphrase variation along two axes: (1) keyword variation and (2) structural variation.

4.1 Keyword Similarity S_K

Keyword similarity measures how much core keywords expressing actions and target objects are preserved between an original instruction and its paraphrase. Robotic manipulation instructions are typically structured around explicit actions and their corresponding objects, often following a canonical form such as “[ACT] the [OBJ]” (e.g., “pick up the bowl”). As a result, the intended behavior is often determined by a small set of task-critical tokens rather than by the sentence as a whole.

This property limits the usefulness of form-based NLP metrics for paraphrased robotic instructions. For example, n-gram metrics such as BLEU (Papineni et al., 2002) emphasize lexical overlap and may fail to distinguish paraphrases that preserve actions and objects but differ in expression, through synonym substitution or word-order variation. Function words can also cause superficial grammatical changes to overly influence similarity scores relative to action- or object-level semantics.

In our setting, the two sentences are given as a paraphrase pair. Thus, rather than reassessing overall semantic equivalence, it is more useful to analyze how task-critical components change. Accordingly, we define a keyword-level similarity that focuses on content words expressing actions and objects excluding function words.

Let $O = \{o_1, \dots, o_n\}$ and $P = \{p_1, \dots, p_m\}$ denote the sets of content words extracted from the original and paraphrased instructions, respectively. Each word is represented by an embedding $e(\cdot)$ obtained from Sentence-BERT (Reimers and Gurevych, 2019). The keyword similarity $S_K(O, P)$ is computed by matching each content word o_i in the original instruction to the most similar word in the paraphrase, measured by cosine similarity, and averaging over all o_i :

$$S_K(O, P) = \frac{1}{n} \sum_{i=1}^n \max_{j \in \{1, \dots, m\}} \cos(e(o_i), e(p_j)), \quad (1)$$

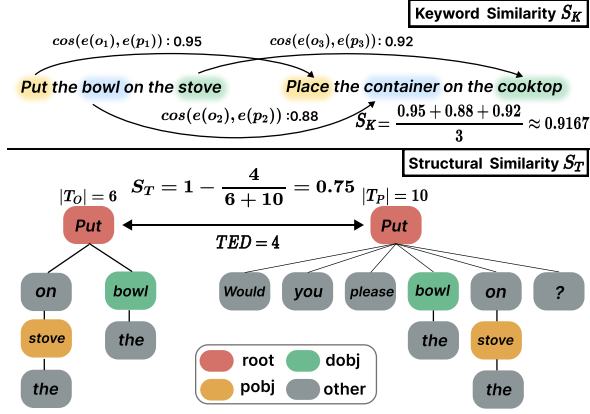


Figure 4: S_K (top) and S_T (bottom) computation. S_K is based on semantic matching between task-critical content words, while S_T uses dependency-tree edit distance. Node colors indicate dependency relations: root (sentence root), dobj (direct object), pobj (object of preposition), and others (remaining types, simplified for visualization; all included in computation).

where $\cos(\cdot, \cdot)$ denotes cosine similarity. Fig. 4 (top) illustrates the computation of S_K . A higher value of $S_K(O, P)$ indicates better preservation of the original instruction’s key content words in the paraphrase. We further discuss a dependency-head-aware extension of S_K in Appendix B.

4.2 Structural Similarity S_T

While keyword similarity S_K captures the preservation of core lexical items, it does not account for syntactic changes. Transformations such as active-passive alternation or clause reordering can substantially alter the form of an instruction while preserving the same keywords. To capture such variation, we introduce structural similarity S_T .

We measure syntactic change using the tree edit distance (TED) (Augsten and Böhlen, 2013) between the dependency trees of the original and paraphrased instructions, denoted by T_O and T_P . TED is defined as the minimum number of edit operations—node and edge insertions, deletions, and substitutions—required to transform one tree into the other. To focus on structural differences, we compute TED on dependency trees whose node labels are part-of-speech (POS) tags and whose edge labels are dependency relations rather than surface words, reducing sensitivity to lexical substitutions.

To mitigate sentence-length effects, we normalize TED by the combined size of two trees and define structural similarity $S_T(T_O, T_P)$ as follows:

$$S_T(T_O, T_P) = 1 - \frac{\text{TED}(T_O, T_P)}{|T_O| + |T_P|}. \quad (2)$$

where $|\cdot|$ denotes the number of nodes in tree T . Fig. 4 (bottom) illustrates the computation of S_T . Lower values of $S_T(T_O, T_P)$ indicate greater structural divergence, such as word-order changes or reorganization of modification relations. Additional validation of dependency-parser stability and the choice of TED normalization is provided in Appendix B.

4.3 PRIDE Score

Robustly assessing paraphrased robotic manipulation instructions requires considering both (i) whether task-critical action/object keywords are preserved (S_K) and (ii) how much the imperative structure is altered (S_T). We verify that the two components indeed measure distinct dimensions, each correlating consistently with independent lexical and syntactic measures of its own axis (Appendix B). Accordingly, we define Paraphrase Distance (PD) by combining S_K and S_T to quantify the overall deviation between an original robotic instruction and its paraphrase:

$$\text{PD} = 1 - \left(\alpha S_K(O, P) + (1 - \alpha) S_T(T_O, T_P) \right), \quad (3)$$

where $\alpha \in [0, 1]$ is a diagnostic weighting factor that controls the relative emphasis on keyword and structural similarity, rather than a hyperparameter to be tuned for performance. We use $\alpha = 0.5$ as a neutral default that weights the two components equally; varying α exposes whether a model is more sensitive to lexical or structural variation. Higher PD indicates greater semantic and structural deviation. Combining PD with task success yields the PRIDE score:

$$\text{PRIDE} = \begin{cases} \text{PD}, & \text{success,} \\ 0, & \text{failure.} \end{cases} \quad (4)$$

PRIDE is a success-conditioned robustness score. A failed sample receives zero robustness credit, while PD, defined for every instruction regardless of task success, continues to represent its linguistic difficulty. Assigning zero to failures therefore does not discard difficulty information, but reflects that the model failed to handle the corresponding paraphrase. PRIDE thus complements binary success metrics by distinguishing whether a VLA model can succeed under paraphrased instructions that exhibit larger semantic and structural deviations.

Method	LIBERO-Goal SR	LIBERO-Para SR (Drop)
OpenVLA-OFT _{goal}	97.9	64.7 (-33.2)
OpenVLA-OFT _{mixed}	96.1	63.7 (-32.4)
$\pi_{0.5}$	97.6	71.4 (-26.2)
$\pi_{0.5}$ (expert-only)	78.6	39.1 (-39.5)
X-VLA	97.8	62.1 (-35.7)
VLA-Adapter	98.2	46.3 (-51.9)
Xiaomi-Robotics-0	98.8	76.0 (-22.8)

Table 2: Success rate (SR) comparison between LIBERO-Goal and LIBERO-Para. Drop denotes the absolute decrease in success rate.

5 Experiment

5.1 Setup

We evaluate seven model configurations (0.6B–7.5B) spanning four architecture families: parallel decoding with action chunking (OpenVLA-OFT), VLMs with a flow-matching action expert ($\pi_{0.5}$, Xiaomi-Robotics-0), soft-prompted cross-embodiment (X-VLA), and bridge-based adaptation (VLA-Adapter). Within the same architecture, we include controlled comparisons on fine-tuning data scope (OFT_{goal} vs. OFT_{mixed}) and VLM training strategy ($\pi_{0.5}$ full vs. expert-only). Full specifications are in Appendix C.1.

5.2 Results

Success Rate Comparison. Tab. 2 compares success rates between LIBERO-Goal and LIBERO-Para. All models exhibit consistent performance degradation ranging from 22.8 pp to 51.9 pp, indicating that the effect is not architecture-specific but pervasive across models. Even the top-performing models on LIBERO-Goal (Xiaomi-Robotics-0: 98.8%, VLA-Adapter: 98.2%) suffer drops of 22.8 pp and 51.9 pp under paraphrasing, respectively, with VLA-Adapter losing nearly half of its performance.

PRIDE Reveals Hidden Severity. Uniform SR treats all paraphrases equally, assigning the same reward to easy and difficult variations, and thus cannot distinguish success limited to easy paraphrases from success that generalizes to harder ones. PRIDE mitigates this limitation by weighting rewards by difficulty. Tab. 3 re-evaluates the same results under PRIDE. VLA-Adapter (22.0%) and $\pi_{0.5}$ expert-only (18.2%) show large drops relative to SR, indicating success mainly on easy paraphrases and systematic failures on harder variations. In contrast, $\pi_{0.5}$ (8.4%) and Xiaomi-Robotics-0

Method	SR	PRIDE	Overestimation (%)
VLA-Adapter	46.3	36.1	22.0
$\pi_{0.5}$ (expert-only)	39.1	32.0	18.2
X-VLA	62.1	52.7	15.1
OpenVLA-OFT _{mixed}	63.7	56.3	11.6
OpenVLA-OFT _{goal}	64.7	58.8	9.1
Xiaomi-Robotics-0	76.0	69.2	8.9
$\pi_{0.5}$	71.4	65.4	8.4

Table 3: SR and PRIDE scores on LIBERO-Para, sorted by overestimation. Overestimation is computed as (SR – PRIDE) / SR, indicating how much uniform success rate overstates a model’s paraphrase robustness.

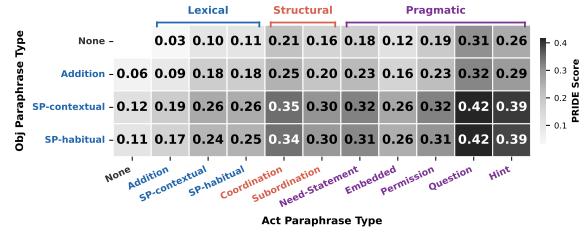


Figure 5: Average PRIDE score per Object $\times$ Action cell in LIBERO-Para (darker = harder). Scores increase along both axes, with the most indirect action types (Question, Hint) combined with object paraphrasing reaching the highest (SP-habitual $\times$ Question: 0.42).

(8.9%) exhibit lower overestimation, showing more uniform robustness. PRIDE also predicts success rate more faithfully than general-purpose NLP metrics such as METEOR (Banerjee and Lavie, 2005), BERTScore, and BLEU, achieving the strongest correlation with SR (Pearson $|r| = 0.802$) and ranking first on 6 of 7 VLAs: the sole exception is the model whose low base-task performance confounds the correlation (Tab. 13, Appendix B).

Fig. 5 and Fig. 6 confirm these trends at cell level: degradation intensifies along both axes, with the sharpest drops when object paraphrasing combines with indirect actions. Notably, the gap between object-preserved rows (None, Addition) and object-paraphrased rows (SP-contextual, SP-habitual) is substantially larger than the action-type gap within the same object condition, suggesting that object-level variation is a stronger driver of failure than action indirectness. Beyond this asymmetry, the two axes do not act independently. Tab. 4 decomposes success rates by cell partition: varying both axes jointly degrades performance by 42.6 pp, exceeding the sum of the individual axis effects (12.9 + 16.4 = 29.3 pp). This super-additive interaction is inaccessible to single-axis paraphrasing. We investigate the object–action asymmetry and its under-

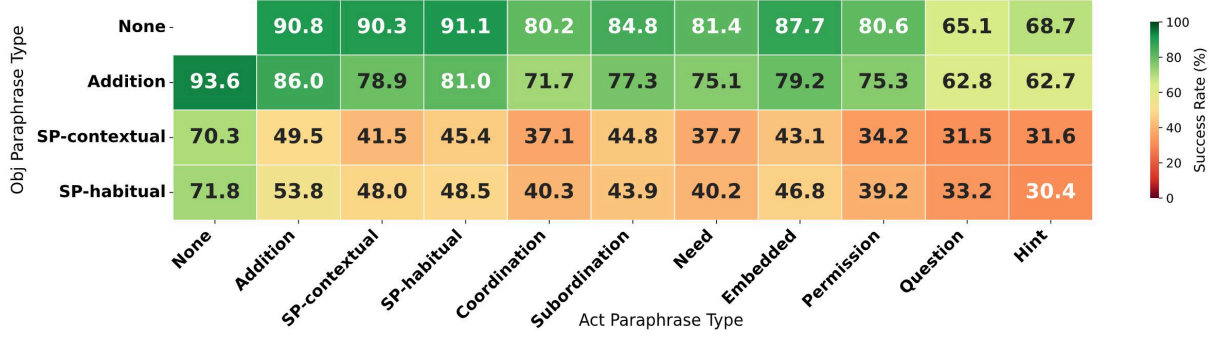


Figure 6: Model-average success rate per Object $\times$ Action cell. Object-paraphrased rows drop sharply compared to object-preserved rows, reaching 30.4% at SP-habitual $\times$ Hint.

Cell group	n	Mean SR (%)	Drop
Original (None $\times$ None)	1	95.0	—
Action-only (Obj: None)	10	82.1	−12.9
Object-only (Act: None)	3	78.6	−16.4
Compositional	30	52.4	−42.6

Table 4: Success rate (SR) by cell group (n : number of cells), averaged over 7 VLAs. Original from Tab. 2, others aggregated from Fig. 6. Drop is relative to Original.

lying causes in Sec. 6.2–6.3, after first examining whether architecture or training choices mitigate the overall degradation (Sec. 6.1).

6 Analysis

6.1 Finding 1: Paraphrase Fragility Persists Across Architectures, Data Scales, and Fine-tuning Strategies

Before analyzing where and how failures occur in Sec. 6.2 and 6.3, we examine whether paraphrase fragility can be attributed to specific factors by varying architecture family, training data scope, and VLM fine-tuning strategy.

Architecture Diversity. Across seven configurations spanning four architecture families (OpenVLA-OFT, $\pi_{0.5}$ /Xiaomi-Robotics-0, X-VLA, VLA-Adapter), all models show substantial success rate drops under paraphrasing, ranging from 22.8 pp to 51.9 pp (Tab. 2). The 7.5B OpenVLA-OFT shows PRIDE comparable to the 0.9B X-VLA. All models exhibit PRIDE overestimation of 8.4–22.0% (Tab. 3). Overall, VLAs experience significant degradation under paraphrasing, regardless of architecture or scale.

Data Scope. OpenVLA-OFT_{mixed} expands task-level data diversity by 4 $\times$ compared to OpenVLA-OFT_{goal} within the same architecture and simulator. However, both models exhibit similar success

rate drops on LIBERO-Para (32.4 pp vs. 33.2 pp), suggesting that increasing task diversity through additional training samples does not improve robustness to linguistic variation in learned tasks.

VLM Training Strategy. We compare the standard $\pi_{0.5}$ model (jointly fine-tuning the VLM and Action Expert) with a variant that freezes the VLM of the base VLA and fine-tunes only the Action Expert. The frozen-VLM variant shows substantially lower performance on LIBERO-Goal (97.6 $\rightarrow$ 78.6; Tab. 2) and does not exhibit improved robustness on LIBERO-Para (SR: 39.1, PRIDE: 32.0; Tab. 3). Although the jointly fine-tuned model achieves higher success rates on LIBERO-Para (71.4 vs. 39.1), both variants still show substantial drops under paraphrasing. This suggests that joint adaptation of the VLM and Action Expert is essential for downstream performance, while fine-tuning on limited demonstrations may degrade pretrained semantics, causing paraphrase vulnerability.

Taken together, paraphrase fragility persists across all three factors. This indicates that the robustness gap cannot be explained solely by architecture, data scope, or fine-tuning strategy, but points to a deeper challenge. *Which linguistic variations, then, are most responsible for these failures?*

6.2 Finding 2: Object Grounding Emerges as a Primary Bottleneck

Sec. 6.1 shows that paraphrase fragility persists regardless of architecture, data scope, or fine-tuning strategy. We next examine where this degradation concentrates. Our analysis reveals an asymmetry: object-level variation emerges as the dominant source of failure, while action indirectness introduces additional degradation. Fig. 7 compares success rates between object-preserved and object-paraphrased instructions across mod-

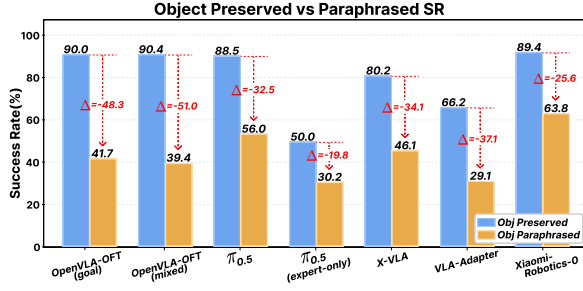


Figure 7: Success rate comparison between object-preserved (None, Addition) and object-paraphrased (SP-contextual, SP-habitual) instructions. All models show substantial drops, from 19.8 pp ($\pi_{0.5}$ expert-only) to 51.0 pp (OpenVLA-OFT_{mixed}). Δ annotated per pair.

els. When the object is paraphrased—even through common synonyms such as replacing *stove* with *range*—performance drops by 19.8 pp ($\pi_{0.5}$ expert-only) to 51.0 pp (OpenVLA-OFT_{mixed}) across models. This gap appears consistently across architectures, suggesting that current VLAs rely more on surface-level keyword matching than on semantic understanding of objects. Notably, OpenVLA-OFT_{mixed}, trained with four times more tasks, exhibits nearly the same gap as OpenVLA-OFT_{goal} (51.0 pp vs. 48.3 pp; Fig. 7), indicating that task diversity and object-paraphrase robustness are decoupled. PRIDE α sweep experiments further confirm that keyword-level lexical variation around object references drives most of the degradation, compared to syntactic restructuring (Appendix D.2). In addition, action indirectness introduces a stepwise performance decline: as instructions become less explicit, success rates drop from 82.7% (None) to around 48% (Question, Hint) (see Appendix D.2 for the full action-axis breakdown).

This asymmetry reflects structural properties of tabletop manipulation. The action space is restricted to a small set of motor primitives (e.g., pick, place, push and open), and each object typically supports only a few feasible actions (e.g., *stove* $\rightarrow$ turn on), allowing models to converge to the correct primitive even under varied phrasing. In contrast, the object space is much larger and lexically open-ended, concentrating combinatorial complexity on object references. This vulnerability may be amplified by current VLA training data, where objects are often referred to by a single canonical name (Fig. 12), making grounding sensitive even to simple synonym substitutions. These observations suggest that the primary bottleneck in paraphrase robustness lies in object grounding, with

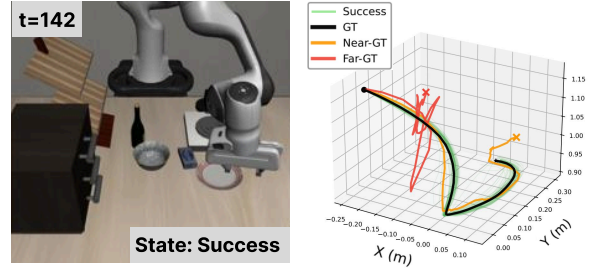


Figure 8: (Left) LIBERO scene for Task 2: Push the plate to the front of the stove. (Right) 3D end-effector trajectories under a paraphrased instruction ($\pi_{0.5}$). Green: successful episodes; black: their mean (GT); orange: Near-GT failure (tracks GT but fails); red: Far-GT failure (diverges early).

Model	Success	Failure		
	Rate	Near-GT	Far-GT	Far-GT(%)
OpenVLA-OFT _{goal}	64.7	1.6	33.7	95.5
Xiaomi-Robotics-0	76.0	1.8	22.2	92.5
VLA-Adapter	46.3	4.2	49.5	92.2
$\pi_{0.5}$	71.4	2.4	26.2	91.6
OpenVLA-OFT _{mixed}	63.7	3.3	33.0	90.9
X-VLA	62.1	5.2	32.7	86.3
$\pi_{0.5}$ (expert-only)	39.1	12.5	48.4	79.5

Table 5: Failure classification on LIBERO-Para (by Far-GT (%)). Near-GT: execution-level failure near GT trajectory. Far-GT: planning-level failure from GT. Across models, 79.5–95.5% of failures are planning-level.

action indirectness introducing additional degradation. Having identified the dominant factor behind these failures, we next ask: *do these failures arise during execution of the correct task, or do models generate different trajectories from the outset?*

6.3 Finding 3: Failures Are Predominantly Planning-Level, Not Execution-Level

Sec. 6.2 identified object grounding as the primary bottleneck, with action indirectness introducing additional degradation. To determine whether failures arise during execution of the correct task or from generating entirely different trajectories from the outset, we classify failures based on trajectory similarity to successful executions. For each task, we define the mean successful trajectory as pseudo ground-truth (hereafter GT) and compute the Dynamic Time Warping (DTW) distance (Sakoe and Chiba, 1978) between each failure trajectory and the GT. Because LIBERO-Goal training data follows a fixed trajectory with minimal path variation, the mean success trajectory serves as a reliable pseudo GT. Failures within a threshold τ (the maximum DTW distance among successful episodes)

are categorized as **Near-GT** (execution-level), indicating correct task execution but failure due to minor motor control errors. Failures exceeding τ are categorized as **Far-GT** (planning-level), indicating fundamentally different trajectories and thus failure in task identification. As shown in Fig. 8, Near-GT trajectories remain close to successful ones, whereas Far-GT trajectories diverge substantially. Additional methodological details are provided in Appendix D.3.

Tab. 5 summarizes the classification results. Across models, 79.5%–95.5% of failures are Far-GT, while Near-GT cases account for less than 5% in most models. This indicates that under paraphrased instructions, models rarely fail along the correct trajectory but instead generate different trajectories from the outset. The only exception is $\pi_{0.5}$ expert-only (Near-GT: 12.5%), where the frozen VLM may identify the task correctly but the non-adapted Action Expert fails to execute it precisely, consistent with the VLM training strategy analysis in Sec. 6.1. These findings align with Sec. 6.2: when object grounding fails, the model plans actions toward incorrect targets, producing trajectories that diverge from the GT. The dominant failure mode lies in task identification rather than motor control, suggesting that improving paraphrase robustness should focus on instruction semantic-to-task mapping rather than action-execution control.

Cross-Benchmark Validation. Finally, to assess whether our findings generalize beyond LIBERO, we construct CALVIN-Para by applying the same two-axis design to CALVIN (Mees et al., 2021), a long-horizon language-conditioned benchmark with a distinct simulator, task suite, and instruction style. We evaluate two CALVIN-native VLAs, RoboFlamingo (Li et al., 2024) and FLOWER (Reuss et al., 2025), whose architectures differ from all seven LIBERO configurations. We confirm that all three findings consistently hold for these models. Despite near-saturated canonical performance, both models collapse under paraphrasing (92.3%→53.1% and 99.7%→58.3%), extending Finding 1 to two additional architectures. Object-axis paraphrasing degrades SR by 32.6 and 37.7 pp, reproducing the object bottleneck of Finding 2. Likewise, 93.4% and 97.1% of failures are planning-level (Far-GT), matching Finding 3. SR also overstates robustness by 18.1% and 17.2% relative to PRIDE, confirming that PRIDE captures hidden severity. Setup and results are in Appendix E.

7 Conclusion

This work investigates paraphrase robustness in modern VLA models using LIBERO-Para, a controlled benchmark that independently varies action and object expressions, and PRIDE, a fine-grained robustness metric. We find that paraphrase fragility persists across architectures, scales, and fine-tuning strategies, with object-level lexical variation as the dominant source of degradation and 80–96% of failures arising from planning-level trajectory divergence rather than execution errors. All three findings reproduce on CALVIN-Para, indicating they are not artifacts of a single benchmark. These results reveal a fundamental limitation: current VLA models map instructions to task identification by surface-level matching rather than robust object grounding. Improving paraphrase robustness therefore requires prioritizing instruction-to-task identification over control refinement, with object grounding as a key direction.

Limitations

We evaluate VLA models only in the LIBERO simulator, which differs from physical platforms in rendering, physics, and sensor noise, so whether these vulnerabilities persist in the real world is untested. Second, we vary one type per axis at a time, leaving compound variations to future work. Third, we do not study paraphrase-based augmentation, since LLM-generated paraphrases could overlap distributionally with the benchmark. Finally, we do not probe attention or intermediate representations: the evaluated models span four architecture families differing in VLM–action-expert coupling, attention scheme, backbone, and scale (Tab. 18), so single-model probing characterizes that model rather than VLAs broadly. Testing the structural hypothesis of Sec. 6.2 is left to future work.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant (RS-2025-00555943); by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grants (No. RS-2021-II211341; Artificial Intelligence Graduate School Program (Chung-Ang University), RS-2026-25513331; Digital Columbus Project, and IITP-2026-RS-2026-25546026; Leading Generative AI Human Resources Development) funded by the Korea government (MSIT).

References

- Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, and 1 others. 2022. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736.
- Anthropic. 2025. System card: Claude opus 4 & claude sonnet 4. Anthropic system card.
- Nikolaus Augsten and Michael H. Böhlen. 2013. *Similarity Joins in Relational Database Systems*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers.
- Shuai Bai, Qwen Team, and 1 others. 2025. [Qwen3-v1 technical report](#). *arXiv preprint arXiv:2511.21631*.
- Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.
- Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, and 6 others. 2024. [\$\pi_0\$: A vision-language-action flow model for general robot control](#). *arXiv preprint arXiv:2410.24164*.
- Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, and 1 others. 2025. Agibot world colosseum: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*.
- Rui Cai, Jun Guo, Xinze He, Piaopiao Jin, Jie Li, Bingxuan Lin, Futeng Liu, Wei Liu, Fei Ma, Kun Ma, and 1 others. 2026. Xiaomi-robotics-0: An open-sourced vision-language-action model with real-time execution. *arXiv preprint arXiv:2602.12684*.
- Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.
- Susan Ervin-Tripp. 1976. [Is Sybil there? The structure of some American English directives](#). *Language in Society*, 5(1):25–66.
- Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinze He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. 2025. [Libero-plus: In-depth robustness analysis of vision-language-action models](#). *arXiv preprint arXiv:2510.13626*.
- Alvan R Feinstein and Domenic V Cicchetti. 1990. High agreement but low kappa: I. the problems of two paradoxes. *Journal of clinical epidemiology*, 43(6):543–549.
- Zipeng Fu, Tony Z. Zhao, and Chelsea Finn. 2024. [Mobile ALOHA: Learning bimanual mobile manipulation with low-cost whole-body teleoperation](#). *arXiv preprint arXiv:2401.02117*.
- Kilem Li Gwet. 2008. Computing inter-rater reliability and its variance in the presence of high agreement. *British Journal of Mathematical and Statistical Psychology*, 61(1):29–48.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Yuchen Hou and Lin Zhao. 2026. Langgap: Diagnosing and closing the language gap in vision-language-action models. *arXiv preprint arXiv:2603.00592*.
- Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. 2024. Prismatic vlms: Investigating the design space of visually-conditioned language models. In *International Conference on Machine Learning (ICML)*.
- Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, and 1 others. 2024. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*.
- Moo Jin Kim, Chelsea Finn, and Percy Liang. 2025. [Fine-tuning vision-language-action models: Optimizing speed and success](#). *arXiv preprint arXiv:2502.19645*.
- Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. 2024. [Openvla: An open-source vision-language-action model](#). *arXiv preprint arXiv:2406.09246*.
- Venelin Kovatchev, M. Antònia Martí, and Maria Salamó. 2018. [ETPC - a paraphrase identification corpus annotated with extended paraphrase typology and negation](#). In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).
- Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, Hang Li, and Tao Kong. 2024. Vision-language foundation models as effective robot imitators. In *International Conference on Learning Representations (ICLR)*.

- Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. 2023. [Libero: Benchmarking knowledge transfer for lifelong robot learning](#). In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, volume 36, pages 44776–44791.
- Xingchao Liu, Chengyue Gong, and Qiang Liu. 2022. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*.
- Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.
- Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. 2021. [CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks](#). *arXiv preprint arXiv:2112.03227*.
- George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.
- OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Abby O’Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, and 1 others. 2024. 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)*, pages 6892–6903. IEEE.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.
- William Peebles and Saining Xie. 2023. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4172–4182. IEEE.
- Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, and 18 others. 2025. [\$\pi_{0.5}\$: a vision-language-action model with open-world generalization](#). *arXiv preprint arXiv:2504.16054*.
- Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.
- Moritz Reuss, Hongyi Zhou, Marcel Rühle, Ömer Erdiñç Yağmurlu, Fabian Otto, and Rudolf Lioutikov. 2025. Flower: Democratizing generalist robot policies with efficient vision-language-action flow policies. *arXiv preprint arXiv:2509.04996*.
- Hiroaki Sakoe and Seibi Chiba. 1978. [Dynamic programming algorithm optimization for spoken word recognition](#). *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49.
- Stan Salvador and Philip Chan. 2007. Toward accurate dynamic time warping in linear time and space. *Intelligent data analysis*, 11(5):561–580.
- Andreas Steiner, André Susano Pinto, Michael Tschannen, Daniel Keysers, Xiao Wang, Yonatan Bitton, Alexey Gritsenko, Matthias Minderer, Anthony Sherbondy, Shangbang Long, Siyang Qin, Reeve Ingle, Emanuele Bugliarello, Sahar Kazemzadeh, Thomas Mesnard, Ibrahim Alabdulmohsin, Lucas Beyer, and Xiaohua Zhai. 2024. [Paligemma 2: A family of versatile vlms for transfer](#). *arXiv preprint arXiv:2412.03555*.
- Qwen Team. 2024. [Qwen2.5 technical report](#). *arXiv preprint arXiv:2412.15115*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. [Llama 2: Open foundation and fine-tuned chat models](#). *arXiv preprint arXiv:2307.09288*.
- Guodong Wang, Chenkai Zhang, Qingjie Liu, Jinjin Zhang, Jiancheng Cai, Junjie Liu, and Xinmin Liu. 2026. [Libero-x: Robustness litmus for vision-language-action models](#). *arXiv preprint arXiv:2602.06556*.
- Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024a. Multilingual e5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672*.
- Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, and 1 others. 2025. Vla-adapter: An effective paradigm for tiny-scale vision-language-action model. *arXiv preprint arXiv:2509.09372*.
- Zhijie Wang, Zhehua Zhou, Jiayang Song, Yuheng Huang, Zhan Shu, and Lei Ma. 2024b. [LADEV: A language-driven testing and evaluation platform for vision-language-action models in robotic manipulation](#). *arXiv preprint arXiv:2410.05191*.
- Jimmy Wu, Rika Antonova, Adam Kan, Marion Lepert, Andy Zeng, Shuran Song, Jeannette Bohg, Szymon Rusinkiewicz, and Thomas Funkhouser. 2023. [Tidybot: Personalized robot assistance with large language models](#). *arXiv preprint arXiv:2305.05658*.

- Kun Wu, Chengkai Hou, Jiaming Liu, Zhengping Che, Xiaozhu Ju, Zhuqin Yang, Meng Li, Yinuo Zhao, Zhiyuan Xu, Guang Yang, and 1 others. 2024. Robomind: Benchmark on multi-embodiment intelligence normative data for robot manipulation. *arXiv preprint arXiv:2412.13877*.
- Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, and Lu Yuan. 2024. Florence-2: Advancing a unified representation for a variety of vision tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4818–4829.
- Yajat Yadav, Zhiyuan Zhou, Andrew Wagenmaker, Karl Pertsch, and Sergey Levine. 2025. [Robust finetuning of vision-language-action robot policies via parameter merging](#). *arXiv preprint arXiv:2512.08333*.
- Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.
- Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, and 1 others. 2025. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. *arXiv preprint arXiv:2510.10274*.
- Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. 2025. [LIBERO-PRO: Towards robust and fair evaluation of vision-language-action models beyond memorization](#). *arXiv preprint arXiv:2510.03827*.
- Brianna Zitkovich and 1 others. 2023. [RT-2: Vision-language-action models transfer web knowledge to robotic control](#). In *Proceedings of The 7th Conference on Robot Learning (CoRL)*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183.

APPENDIX

Contents

A	LIBERO-Para: A Controlled VLA Benchmark for Paraphrase Robustness	14
A.1	Excluded Types from Extended Paraphrase Typology	14
A.2	Excluded Type from Directive Types	15
A.3	Paraphrase Dataset Generation	15
A.4	Statistics of LIBERO-Para	16
A.5	Human Evaluation	17
B	PRIDE: Paraphrase Robustness Index in Robotic Instructional DEviation	17
C	Experiment	20
C.1	Setup	20
C.2	Result	21
D	Analysis	21
D.1	Finding 1: Paraphrase Fragility Persists Across Architectures, Data Scales, and Fine-tuning Strategies	21
D.2	Finding 2: Object Grounding Is the Primary Bottleneck	21
D.3	Finding 3: Failures Are Predominantly Planning-Level, Not Execution-Level	23
E	CALVIN-Para: Cross-Benchmark Validation	25
E.1	Setup	25
E.2	Results	27
F	AI Assistants	28

Category	Type	Source (Year)
Obj-Lexical	same polarity habitual	EPT (2018)
	Same polarity contextual Addition	
Act-Lexical	Same polarity habitual	EPT (2018)
	Same polarity contextual Addition	
Act-Structural	Coordination Subordination	EPT (2018)
Act-Pragmatic	Personal need	Directive Types (1976)
	Question directive	
	Embedded imperative	
	Permission Hint	

Table 6: Selected paraphrase types in LIBERO-Para. Types are derived from Extended Paraphrase Typology (EPT) (Kovatchev et al., 2018) and Directive Types (Ervin-Tripp, 1976).

A LIBERO-Para: A Controlled VLA Benchmark for Paraphrase Robustness

This appendix details the paraphrase taxonomy adopted in LIBERO-Para and explains the rationale for excluding certain types from the source taxonomies (EPT and Directive Types). Our taxonomy is grounded in the Extended Paraphrase Typology (EPT) (Kovatchev et al., 2018) and the Directive types proposed by Ervin-Tripp (1976). From the 26 atomic paraphrase types in EPT and the six different types in Directive Types, we select 13 types that satisfy the following criteria: (i) applicability to robotic manipulation instructions (i.e., direct imperatives), (ii) preservation of the original meaning, (iii) compliance with visual and spatial constraints, and (iv) grammatical naturalness. All paraphrases are generated under these constraints and are used exclusively for evaluation.

A.1 Excluded Types from Extended Paraphrase Typology

While EPT provides a broad inventory of paraphrase operations, many types are unsuitable for robotic manipulation instructions under our design constraints. Tab. 6 lists the types selected for LIBERO-Para, and Tab. 7 presents the full Extended Paraphrase Typology (EPT) with its high-level categories and corresponding atomic types. Below we describe, category by category, which types we exclude and why.

Morphology. We exclude the entire Morphology category. Inflectional changes may alter object cardinality (e.g., pluralization) or modify temporal interpretation when applied to actions. Modal verb changes can shift intent, introducing seman-

Categories	Atomic Type
Morphology	Inflectional changes
	Modal verb changes
	Derivational changes
Lexicon	Spelling changes
	Same polarity substitution (habitual)
	Same polarity substitution (contextual)
	Same polarity substitution (named entities)
Lexical-syntactic	Change of format
	Opposite polarity substitution (habitual)
	Opposite polarity substitution (contextual)
	Synthetic/analytic substitution
Syntax	Converse substitution
	Diathesis alternation
	Negation switching
	Ellipsis
Discourse	Coordination changes
	Subordination and nesting changes
	Punctuation changes
	Direct/indirect style alternations
Other	Sentence modality changes
	Syntax/discourse structure changes
	Addition/Deletion
Extremes	Change of order
	Semantic based
	Identity
	Non-paraphrase
	Entailment

Table 7: Extended Paraphrase Typology (EPT) categories and atomic types (Kovatchev et al., 2018).

tic drift. Derivational changes alter part-of-speech (e.g., “*pick*” → “*picker*”), which violates imperative structure or disrupts the intended reference.

Lexicon. We retain same polarity substitution (habitual and contextual) and exclude the remaining types. Spelling changes are insufficient to constitute meaningful variation. Same polarity substitution involving named entities is rarely applicable, as robotic instructions predominantly use common nouns and generic verbs. Change of format is often either trivial or difficult to apply while preserving meaning.

Lexical-syntactic. We exclude this category in its entirety. Opposite polarity substitution is unnatural for object nouns and leads to awkward or unintended speech acts when applied to actions. Synthetic/analytic substitutions (e.g., “*bowl*” ↔ “*round container*”) are unnatural in concise imperatives. Converse substitutions introduce role-swapping constructions that are rarely natural in commands.

Syntax. We retain coordination and subordination changes and exclude the remaining types. Diathesis alternation yields passive-like commands, which are unnatural in robotic instructions. Negation switching overlaps with opposite polarity sub-

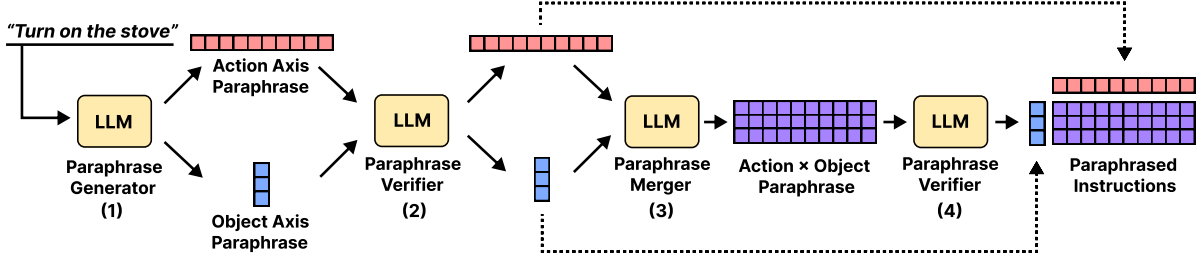


Figure 9: Overview of the LIBERO-Para dataset generation workflow. The process consists of four stages: (1) axis-wise paraphrase generation, (2) verification, (3) merging, and (4) final verification.

Directive Type	Example
Need statements	"I need a match"
Imperatives	"Gimme a match", "a match"
Embedded imperatives	"Could you gimme a match?"
Permission directives	"May I have a match?"
Question directives	"Gotta match?"
Hints	"The matches are all gone"

Table 8: Six directive types from Directive Types (Ervin-Tripp, 1976).

stitutions. Ellipsis introduces ambiguity in short imperatives and overlaps with addition/deletion.

Discourse. We exclude this category. Robotic commands are treated as direct imperatives; alternations in style or sentence modality may alter the intended directive force. Syntax/discourse structure changes are overly high-level relative to atomic instructions and hinder controlled evaluation.

Other. We retain only addition. Given the brevity of imperative commands, deletion frequently removes essential components or produces ungrammatical outputs. Change of order is often unnatural in short imperatives. Semantic-based types lack a precise definition and are unsuitable for controlled evaluation.

Extremes. We exclude this category entirely. Identity involves no transformation. Non-paraphrase violates meaning preservation. Entailment represents an inferential relation rather than a meaning-preserving transformation.

A.2 Excluded Type from Directive Types

Ervin-Tripp proposed a directive taxonomy that categorizes Directive Types into six types (Ervin-Tripp, 1976), as summarized in Tab. 8. Among these, we select five types for the Action-Pragmatic axis: need statements, embedded imperatives, permission directives, question directives, and hints.

Imperatives. We exclude the imperative type from our paraphrase taxonomy. Since imperatives (e.g., "Pick up the bowl") represent the canon-

ical form of robotic manipulation instructions, they serve as the original instruction rather than a paraphrase variant. In our benchmark design, this type corresponds to the baseline condition (Action axis: None) against which other pragmatic variations are compared.

A.3 Paraphrase Dataset Generation

This section describes the paraphrase dataset generation process using LLMs. As illustrated in Fig. 9, our workflow consists of four stages: (1) axis-wise paraphrase generation, (2) paraphrase verification, (3) axis merging, and (4) final verification. Each stage is executed by a role-assigned LLM (generator, merger, or verifier); the full prompts, including the persona, task specification, and output format for every role, are provided in Figs. 22 to 30.

Axis-wise Paraphrase Generation. Given an original instruction, a paraphrase generator (LLM) independently produces paraphrases along the action axis (10 types) and the object axis (3 types). Each generated paraphrase is filtered by a paraphrase verifier (LLM) to ensure meaning preservation and grammatical naturalness.

Paraphrase Merging. Verified action-axis and object-axis paraphrases modify independent components of the instruction and can therefore be combined. If n action paraphrases and m object paraphrases pass verification, up to $n \times m$ merged paraphrases are possible. Merged paraphrases are further validated by the verifier (LLM) before inclusion in the dataset.

Design Principles. Rather than prompting a single LLM to generate all paraphrase types jointly, we adopt an axis-wise generation and merging strategy. This modular design assigns a single role to each LLM (generator, merger, and verifier), reducing task complexity and improving generation reliability. All LLM calls use Gemini 2.5 Pro. Detailed prompts used at each stage are provided at the end

Object	None	Act-Lexical			Act-Structural		Act-Pragmatic					Total
		add	ctx	hab	coord	subord	need	embed	perm	quest	hint	
None	–	100	79	74	98	75	93	93	83	87	88	870
Addition	98	100	100	100	100	100	100	99	99	99	100	1,095
Contextual	87	100	100	100	100	99	100	100	100	94	96	1,076
Habitual	74	100	98	100	97	94	100	95	100	95	98	1,051
Total	259	400	377	374	395	368	393	387	382	375	382	4,092

Abbreviations: add = addition, ctx = same_polarity_contextual, hab = same_polarity_habitual, coord = coordination, subord = subordination, need = need_statement, embed = embedded_imperative, perm = permission_directive, quest = question_directive.

Table 9: LIBERO-Para dataset statistics. Each cell shows the number of paraphrased instructions for the corresponding Object (row) and Action (column) variation type combination. “None” indicates no variation on that axis.

Original Instruction	Count
<i>Put the wine bottle on top of the cabinet</i>	423
<i>Open the middle drawer of the cabinet</i>	416
<i>Turn on the stove</i>	414
<i>Put the wine bottle on the rack</i>	413
<i>Put the cream cheese in the bowl</i>	411
<i>Open the top drawer and put the bowl inside</i>	410
<i>Put the bowl on top of the cabinet</i>	410
<i>Push the plate to the front of the stove</i>	406
<i>Put the bowl on the stove</i>	403
<i>Put the bowl on the plate</i>	386
Total	4,092

Table 10: Number of paraphrased instructions per original instruction in LIBERO-Para.

of the paper for readability, and are illustrated in Figs. 22 to 30.

A.4 Statistics of LIBERO-Para

LIBERO-Para consists of 4,092 paraphrases generated from 10 original LIBERO-Goal instructions, selected from the four LIBERO task types (Spatial, Object, Goal, and Long) where linguistic understanding is essential for successful execution.

The dataset is organized along two axes: an Object axis with three lexical types, and an Action axis with ten types (three lexical, two structural, and five pragmatic). This two-axis design yields 43 distinct paraphrase type combinations: three Object-only, ten Action-only, and thirty compositional types (3×10).

Tab. 9 reports the number of paraphrases for each Object $\times$ Action combination. The dataset includes 259 Object-only paraphrases (Action = None), 870 Action-only paraphrases (Object = None), and 2,963 compositional paraphrases. To facilitate diverse analyses, samples are distributed approximately uniformly across cells, with around 100 paraphrases per cell.

To examine how each component of PRIDE contributes to paraphrase difficulty, Figs. 10 and 11

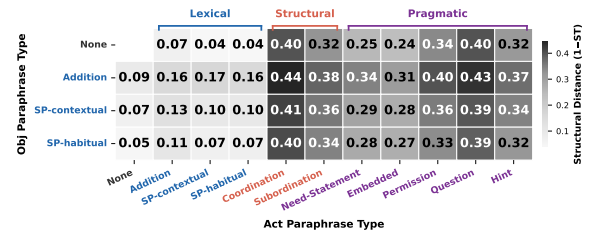


Figure 10: Average Structural Distance ($1 - S_T$) per Object $\times$ Action cell (SK weight = 0.0, ST weight = 1.0 in PRIDE). Structural distance is dominated by action paraphrase type rather than object substitution: Coordination and Subordination columns score above 0.28 across all rows, while lexical action types remain below 0.17, confirming that structural rewriting originates from act-level transformations. Near-zero values are clamped to 0.00 for display.

present the average structural distance ($1 - S_T$) and keyword distance ($1 - S_K$), respectively, which correspond to isolating each term in the PD formulation (Eq. 3). Structural distance is dominated by the Action axis: coordination and subordination columns consistently score above 0.28 regardless of object type, whereas lexical action types stay below 0.17. Keyword distance, in contrast, is primarily driven by the Object axis: contextual and habitual substitutions yield high distances (0.41–0.45) due to synonym replacement, while rows without object paraphrasing remain near zero. This decomposition confirms that the two PRIDE components capture complementary sources of difficulty—syntactic divergence from action paraphrasing and lexical divergence from object paraphrasing.

Finally, Tab. 10 reports the number of paraphrases per original instruction. Each instruction yields 386–423 paraphrases, indicating a balanced distribution.

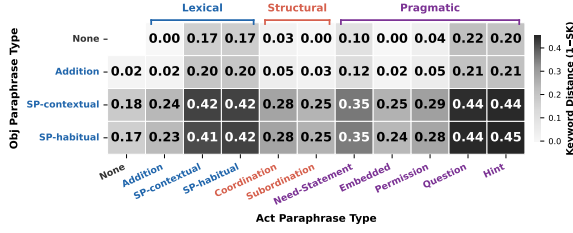


Figure 11: Average Keyword Distance ($1 - S_K$) per Object $\times$ Action cell (SK weight = 1.0, ST weight = 0.0 in PRIDE). Scores are driven primarily by object paraphrasing: SP-contextual and SP-habitual rows score higher regardless of action type, with SP-habitual $\times$ Hint reaching 0.45. Near-zero values are clamped to 0.00 for display.

A.5 Human Evaluation

To verify the semantic validity of LIBERO-Para, we conducted a human evaluation on a randomly sampled 5% subset (205 samples) of the full benchmark. Fifteen annotators independently judged whether each original–paraphrase pair would elicit the same successful behavior in the given scene, using a binary Yes/No decision.

Inter-Annotator Agreement. We report Gwet’s AC1 (Gwet, 2008) as the inter-annotator agreement (IAA) metric. We chose AC1 over Cohen’s or Fleiss’ κ because our labels are heavily skewed toward the positive class, a setting in which κ is known to be substantially deflated despite high observed agreement (Feinstein and Cicchetti, 1990). On our 15-annotator evaluation, Gwet’s AC1 is **0.854**, indicating strong agreement.

Consensus Statistics. Under a majority-vote criterion ($\geq 8/15$ annotators marking Yes), 204 out of 205 samples (99.51%) were judged as meaning-preserving. Under a stricter threshold requiring $\geq 12/15$ agreement (80%), 183 out of 205 samples (89.27%) passed. Across all 205 samples, annotators selected Yes at an average rate of 14.13/15 (94.18%), further supporting high item-level consensus.

Error Analysis. We examined the 22 samples that failed the stricter criterion and found that disagreement was concentrated in paraphrases where the original imperative form was transformed into suggestive, declarative, or indirect speech-act forms. This indicates that disagreement primarily arose from differences in annotator interpretation of speech-act form rather than semantic distortion of the paraphrase itself. At the same time, such

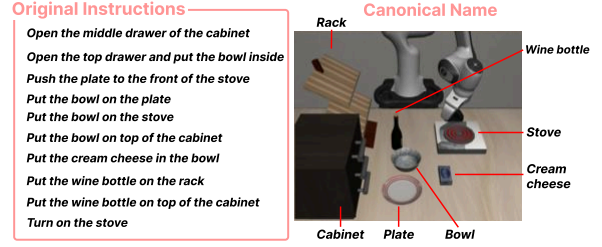


Figure 12: LIBERO-Goal task instructions (left) and corresponding scene with canonical object names (right). Each object is referred to by a single unique keyword throughout all instructions (e.g., *stove*, *bowl*, *rack*), with no lexical variation across tasks.

cases confirm that the benchmark includes pragmatically challenging linguistic reformulations that go beyond simple lexical substitution.

TED Norm.	Pearson	Spearman	Range	[0, 1] Bounded
Sum (Current)	-0.802	-0.831	[0, 1]	Inherently
Max (raw)	-0.715	-0.747	[-1, 1]	No
Max (clipped)	-0.715	-0.747	[0, 1]	Post-hoc
Average	-0.650	-0.676	[-1, 1]	No

Table 11: Ablation of TED normalization choices for S_T . Pearson and Spearman correlations are computed between PRIDE and success rate across 43 paraphrase cells and seven VLA configurations. Max (raw) produces 9 negative values among 4,092 paraphrases, while Max (clipped) applies post-hoc clipping to [0, 1].

Annotation Protocol. Each annotator received an Excel spreadsheet containing 205 randomly sampled original–paraphrase pairs. They were instructed to mark 0 if the paraphrased instruction would elicit the same successful behavior as the original instruction in the given VLA scene (LIBERO-Goal initial scene), and X otherwise. The 15 annotators included participants with varying levels of familiarity with robotic manipulation tasks, ranging from domain-familiar researchers to non-expert volunteers. All annotators were informed that their responses would be used for research purposes.

B PRIDE: Paraphrase Robustness Index in Robotic Instructional DEviation

Motivation. General-purpose NLP distance metrics such as BERTScore (Zhang et al., 2019), BLEU (Papineni et al., 2002), and METEOR (Banerjee and Lavie, 2005) are designed to measure surface-level or semantic similarity between text pairs, without considering how linguistic changes affect downstream task execution. In grounded robotic instruction following, however,

Original	Paraphrase	Type	SR(%)	PRIDE	$1 - S_K$	$1 - S_T$	1-BERT	1-BLEU	1-METEOR
Put the cream cheese in the bowl	"carefully put the cream cheese in the bowl"	addition(act)	90.8	0.03	0.00	0.07	0.14	0.16	0.02
	"put the cheese spread in the vessel"	SP-contextual(obj)	70.3	0.35	0.27	0.43	0.12	0.80	0.36
	"Is the spread supposed to go in the container?"	SP-contextual(obj) & question	31.5	0.56	0.60	0.53	0.28	0.91	0.65
Turn on the stove	"carefully turn on the stove"	addition(act)	90.8	0.06	0.00	0.11	0.15	0.33	0.03
	"turn on the range"	SP-habitual(obj)	71.8	0.36	0.34	0.38	0.19	0.41	0.26
	"Is the range hot yet?"	SP-habitual(obj) & hint	33.2	0.65	0.70	0.60	0.40	0.92	0.88
Open the middle drawer of the cabinet	"carefully open the middle drawer of the cabinet"	addition(act)	90.8	0.03	0.00	0.05	0.10	0.12	0.01
	"Find the cabinet, then proceed to open the middle drawer"	coordination	80.2	0.25	0.00	0.50	0.10	0.12	0.01
	"The storage unit's middle compartment is currently shut"	SP-habitual(obj) & hint	31.6	0.46	0.41	0.50	0.19	0.72	0.49

Table 12: Qualitative comparison of PRIDE, a task-grounded paraphrase distance metric, with general-purpose NLP distance metrics on selected LIBERO-Para examples. Each task group presents three paraphrases of increasing linguistic distance: a minor addition, a lexical substitution, and a compound paraphrase combining object substitution with an indirect speech act. PRIDE increases monotonically as success rate (SR) degrades, reflecting its decomposition into keyword similarity (S_K) and structural similarity (S_T). In contrast, $1 - \text{BERT}$ lacks discriminative range, $1 - \text{BLEU}$ fluctuates inconsistently, and $1 - \text{METEOR}$ fails to capture structurally induced difficulty when keywords are preserved (e.g., coordination in the third group scores 0.01 despite a 10.6 pp SR drop).

not all lexical changes are equally disruptive: replacing a task-critical object noun (e.g., “stove” $\rightarrow$ “range”) directly impacts visual grounding and action selection, whereas syntactic additions (e.g., prepending “carefully”) leave the core command intact. PRIDE is designed to reflect this asymmetry by decomposing paraphrase distance into two robot-relevant axes: keyword divergence (S_K), which captures whether task-critical referents are preserved, and structural divergence (S_T), which measures how far the utterance departs from the imperative form that VLA models are predominantly trained on.

Dependency-Head-Aware Extension of S_K . The default S_K measures action/object-centered lexical preservation using content-word matching. While this formulation is suitable for the short and structured instructions considered in our benchmark, it does not explicitly distinguish task-critical action/object heads from peripheral modifiers.

As an extension, dependency parsing can be used to isolate action roots, direct objects, and nominal modifiers, enabling head-only or head-weighted variants of S_K . For example, given “*Put the bowl on the stove*” and “*Place the container on the cooktop*,” a head-aware variant can explicitly compare *Put* $\leftrightarrow$ *Place* as action heads and *bowl* $\leftrightarrow$ *container* as object heads, while distinguishing such task-critical components from peripheral modifiers. This provides a more targeted view of lexical preservation when dependency heads can be reliably identified. We retain the content-word-based S_K in Eq. (1) as the default formulation; the dependency-head-aware formulation is provided as an extension and is not used to recompute the results reported in this work.

Parser Stability. We use the spaCy `en_core_web_sm` parser to obtain the dependency trees used in S_T . To assess parser stability, we manually verified the dependency assignments for a random sample of 150 paraphrases from LIBERO-Para. Of these, 146 out of 150 parses were judged correct, corresponding to an accuracy of 97.3%. This validation supports the stability of the dependency parsing used to compute S_T across the paraphrases in our benchmark.

TED Normalization Ablation. The normalization in Eq. (2) is designed to keep S_T within the $[0, 1]$ range. For two dependency trees T_O and T_P , TED is upper-bounded by $|T_O| + |T_P|$; therefore, the sum normalization used in Eq. (2) inherently guarantees bounded structural similarity. We additionally compare this choice against max, clipped-max, and average tree-size normalization, as shown in Table 11.

The sum normalization yields the strongest PRIDE–SR correlation while inherently preserving the $[0, 1]$ range. Max and average normalizations divide the same TED value by smaller denominators, which increases the structural penalty and can over-amplify minor syntactic variation. Accordingly, we retain the sum normalization in Eq. (2).

Qualitative Comparison with NLP Metrics. Tab. 12 illustrates how PRIDE captures task-relevant linguistic variation compared to general-purpose NLP metrics. For each task group, we present three paraphrases of increasing difficulty: a minor addition (e.g., prepending “carefully”), a lexical substitution of the object or action, and a compound paraphrase combining object substitution with an indirect speech act. PRIDE increases monotonically as success rate degrades—for instance, in the first group, PRIDE rises from 0.03

to 0.35 to 0.56 as SR drops from 90.8% to 70.3% to 31.5%. This graduated behavior stems from the complementary design of its two components: S_K remains near zero for syntactic-only changes (e.g., addition) but sharply increases when task-critical keywords are replaced, while S_T captures structural divergence from the imperative form even when keywords are preserved.

In contrast, conventional NLP metrics each exhibit notable limitations. 1-BERTScore (Zhang et al., 2019) remains in a narrow range (0.10–0.28) across all paraphrase types, failing to distinguish between benign additions and highly disruptive compound paraphrases. 1-BLEU (Papineni et al., 2002) behaves erratically: in the first group, it assigns a higher distance to a simple object substitution (0.80) than the gap between that substitution and a far more disruptive compound form (0.80 $\rightarrow$ 0.91), compressing meaningful difficulty differences. 1-METEOR (Banerjee and Lavie, 2005) tracks the overall degradation trend more faithfully than the other two metrics, owing to its synonym and stem matching via WordNet (Miller, 1995). However, it still fails to capture structurally induced difficulty: in the third group, coordination (“Find the cabinet, then proceed to open the middle drawer”) receives the same score as a trivial addition (both 0.01), despite a 10.6 pp SR gap (90.8% $\rightarrow$ 80.2%), because the original keywords are largely preserved. More fundamentally, METEOR provides only a single scalar distance and cannot decompose *why* a paraphrase is distant—whether due to keyword replacement or structural transformation—limiting its diagnostic utility. PRIDE addresses this through its explicit S_K/S_T decomposition, enabling researchers to attribute performance degradation to specific linguistic dimensions.

Quantitative Validation. Beyond qualitative examples, we verify that PRIDE correlates with actual task performance. Fig. 16 plots the mean success rate of each paraphrase cell against its PRIDE score for all seven models. All models exhibit statistically significant negative correlations (Pearson r ranging from -0.671 to -0.877 , $p < .0001$), confirming that higher paraphrase distance consistently leads to lower task success. This validates PRIDE as a meaningful difficulty metric for paraphrase robustness evaluation.

Moreover, PRIDE outperforms general-purpose NLP metrics in predicting success rate. As shown

Metric	r	ρ	Ranked 1st (of 7)
PRIDE (ours)	-0.802	-0.831	6
1-METEOR	-0.777	-0.785	1 [†]
1-BERTScore	-0.669	-0.689	0
1-BLEU	-0.642	-0.716	0

Table 13: Correlation with success rate over 43 cells $\times$ 7 VLAs (5 seeds per VLA). r denotes Pearson’s and ρ denotes Spearman’s correlation. Ranked 1st: number of VLAs on which the metric achieves the strongest correlation. All values are significant ($p < 0.0001$). [†]1-METEOR ranks first only on $\pi_{0.5}$ (expert-only), the sole model with base-task SR below 90%, where base-task noise dominates.

Cell group (n)	PRIDE		1-METEOR	
	r	ρ	r	ρ
Action-only (10)	-0.747	-0.733	-0.719	-0.711
Object-only (3) [‡]	-0.969	-0.857	-0.920	-0.857
Compositional (30)	-0.756	-0.799	-0.678	-0.676

Table 14: Per-cell-group correlation with success rate (mean across 7 VLAs, 5 seeds per VLA), following the cell partition of Tab. 4. All values are significant ($p < 0.0001$). [‡]For Object-only ($n = 3$), the small sample limits Spearman resolution: both metrics tie at -0.857 , while Pearson remains discriminative.

in Tab. 13, PRIDE achieves the strongest correlation ($r = -0.802$, $\rho = -0.831$), outperforming 1-METEOR, 1-BERTScore, and 1-BLEU, and ranks first on 6 of 7 VLAs. The sole exception is $\pi_{0.5}$ (expert-only), the only model with base-task SR below 90% (78.6% vs. 96–99% for the others), a degenerate case where base-task noise dominates the signal. Per-seed analysis confirms that this advantage is deterministic at the model level: PRIDE ranks first in all 30 conditions (6 VLAs $\times$ 5 seeds, excluding the degenerate case). Tab. 14 further decomposes this advantage by cell group against the strongest baseline, 1-METEOR. PRIDE leads in every group, and its margin is largest in the Compositional setting: simultaneous action and object substitution saturates METEOR’s unigram matching, while the Sentence-BERT-based S_K and dependency-based S_T continue to resolve both lexical and structural variation.

External Validation of S_K and S_T . To verify that the two components capture disentangled linguistic dimensions, we correlate PD, S_K , and S_T with four independent OOD-ness measures computed by formulations distinct from PRIDE: E5 cosine distance (semantic) (Wang et al.,

OOD measure (category)	PD	S_K	S_T
E5 cosine distance (semantic)	0.844	0.784	0.378*
Content-lemma novelty (lexical)	0.789	0.882	0.189 [^]
Dependency-triple Jaccard (lexical)	0.886	0.841	0.377*
POS-bigram cosine distance (syntactic)	0.628	0.001 [^]	0.921

Table 15: Absolute Pearson correlation $|r|$ between PRIDE components and four independent OOD-ness measures at the 43-cell level (same setting as Fig. 16). Unmarked: $p < 0.0001$; *: $0.0001 \leq p < 0.05$; [^]: not significant ($p \geq 0.05$). S_K aligns with lexical measures but not the syntactic one; S_T shows the opposite pattern.

2024a), content-lemma novelty and dependency-triple Jaccard (lexical), and POS-bigram cosine distance (syntactic). As shown in Tab. 15, S_K aligns strongly with lexical measures ($|r|$ up to 0.882) while remaining uncorrelated with the syntactic one ($|r| = 0.189$, n.s.), and S_T shows the exact opposite pattern ($|r| = 0.921$ syntactic vs. $|r| = 0.001$ lexical, n.s.). This double dissociation demonstrates that S_K and S_T capture complementary, non-overlapping dimensions. Notably, S_K ’s higher correlation with content-lemma novelty than PD is consistent with its design intent: Sentence-BERT embeddings are sensitive to lexical substitutions, aligning S_K with the lexical-content axis. This disentanglement enables the axis-specific diagnosis of VLA failures that single-score metrics cannot provide.

C Experiment

C.1 Setup

Computing Infrastructure. All experiments were conducted on NVIDIA RTX A6000 and NVIDIA L40S GPUs. Specifically, OpenVLA-OFT variants were evaluated on RTX A6000 GPUs, while all other models (X-VLA, VLA-Adapter, $\pi_{0.5}$, $\pi_{0.5}$ (expert-only), and Xiaomi-Robotics-0) were evaluated on L40S GPUs. The total evaluation cost across all seven model configurations amounts to approximately **194 GPU hours** (Tab. 16). For $\pi_{0.5}$ (expert-only), the training was also performed on L40S GPUs following the original $\pi_{0.5}$ fine-tuning protocol (see Tab. 20 for training hyperparameters and Fig. 15 for the training loss curve).

Model Weights and Code. The code repositories and pretrained weights are listed in Tab. 17. All evaluated models use publicly released checkpoints and official codebases, except for $\pi_{0.5}$ (expert-only), which we fine-tuned from the base $\pi_{0.5}$ checkpoint by freezing the VLM and updating only

Model	GPU	VRAM (GB)	Eval Hours
OpenVLA-OFT _{goal}	A6000	~16	~12
OpenVLA-OFT _{mixed}	A6000	~16	~12
X-VLA	L40S	~6.5	~11
VLA-Adapter	L40S	~3	~11
$\pi_{0.5}$	L40S	~38	~70
$\pi_{0.5}$ (expert-only)	L40S	~38	~70
Xiaomi-Robotics-0	L40S	~14	~8
Total			~194

Table 16: Evaluation GPU hours and peak VRAM usage per model configuration.

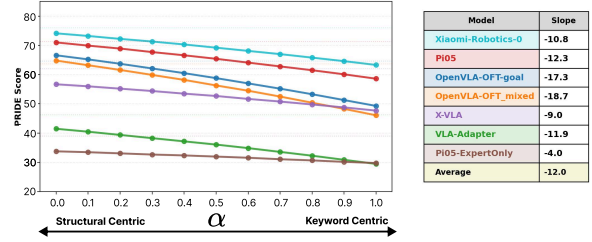


Figure 13: Effect of the weighting parameter α on PRIDE scores across all models. Left: as α increases from 0 (structure-centric) to 1 (keyword-centric), PRIDE scores decrease consistently for all models, indicating that keyword-based evaluation assigns higher credit to samples that models already solve easily. Right: per-model linear slope of the PRIDE- α curve. Steeper negative slopes indicate stronger dependence on keyword similarity over structural similarity.

the action expert. Tab. 18 summarizes architecture-level specifications, and Tab. 19 details the fine-tuning configurations.

Backbone and Data References. The VLM backbones used across evaluated models include Prismatic (Karamcheti et al., 2024) with Llama 2 (Touvron et al., 2023), PaliGemma 2 (Steiner et al., 2024), Florence-2 (Xiao et al., 2024), Qwen2.5 (Team, 2024), and Qwen3-VL (Bai et al., 2025). For pre-training data, OpenVLA-OFT uses the Open X-Embodiment (OXE) dataset (O’Neill et al., 2024), and X-VLA is pre-trained on Droid (Khazatsky et al., 2024), RoboMind (Wu et al., 2024), and Agibot (Bu et al., 2025). All $\pi_{0.5}$ variants use AdamW (Loshchilov and Hutter, 2017) as the optimizer (Tab. 20).

Evaluation Protocol. Each model is evaluated across 5 different random seeds (7, 8, 9, 10, 11) per task-paraphrase configuration. All reported success rates represent the **mean over 5 seeds**; standard deviations are not reported, as our analysis focuses on aggregate robustness trends across paraphrase types rather than per-configuration variance. We

Model	Code	Weights
OpenVLA-OFT _{goal}	https://github.com/moojink/openvla-oft	https://huggingface.co/moojink/openvla-7b-oft-finetuned-libero-goal
OpenVLA-OFT _{mixed}	(same as above)	https://huggingface.co/moojink/openvla-7b-oft-finetuned-libero-spatial-object-goal-10
$\pi_{0.5}$	https://github.com/Physical-Intelligence/openpi (JAX)	gs://openpi-assets/checkpoints/pi05_libero
$\pi_{0.5}$ (expert-only)	(same as $\pi_{0.5}$)	Fine-tuned from gs://openpi-assets/checkpoints/pi05_base
X-VLA	https://github.com/huggingface/lerobot	https://huggingface.co/lerobot/xvla-libero
VLA-Adapter	https://github.com/OpenHelix-Team/VLA-Adapter	https://huggingface.co/VLA-Adapter/LIBERO-Goal-Pro
Xiaomi-Robotics-0	https://github.com/XiaomiRobotics/Xiaomi-Robotics-0	https://huggingface.co/XiaomiRobotics/Xiaomi-Robotics-0-LIBERO

Table 17: Code repositories and pretrained weight sources for all evaluated models.

Model	Release	Arch. Type	VLM Backbone	VLM Params	Action Module	Action Params	Total Params
OpenVLA-OFT	2025.03	Parallel Decoding	Prismatic (Llama 2)	7B	L1 MLP	<1M	7.5B
$\pi_{0.5}$	2025.09	VLM + Action Expert	PaliGemma 2	3B	Flow matching expert	0.3B	3.3B
VLA-Adapter	2025.09	Bridge-based	Prismatic (Qwen2.5-0.5B)	0.5B	Bridge Attention Policy	97M	0.6B
X-VLA	2026.01	Soft-prompted	Florence-2	0.5B	Flow matching transformer	~0.4B	0.9B
Xiaomi-Robotics-0	2026.02	VLM + Action Expert	Qwen3-VL-4B	4B	Flow matching DiT	~0.7B	4.7B

Table 18: Architecture-level specifications of the evaluated VLA models. Release denotes the public code release date (YYYY.MM). Models span a range of architectural paradigms—from parallel decoding to bridge-based adapters to flow matching action experts—with total parameter counts ranging from 0.6B to 7.5B. OpenVLA-OFT variants (goal/mixed) share the same architecture and are listed as a single entry.

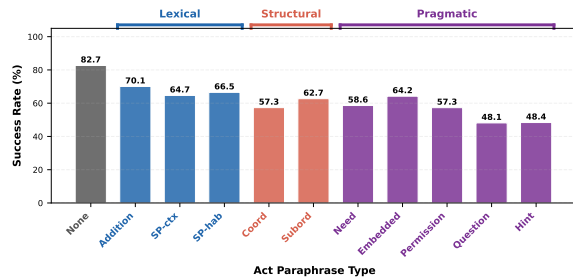


Figure 14: Success rate breakdown by action paraphrase type, averaged across all 7 model configurations. Paraphrase types are grouped into three linguistic categories: **Lexical** (surface-level word changes), **Structural** (syntactic reorganization), and **Pragmatic** (indirect speech acts). Performance degrades progressively from the original instruction (82.7%) through lexical variants (66–70%) and structural variants (57–63%) to the most indirect pragmatic forms such as Question (48.1%) and Hint (48.4%).

use the LIBERO simulation environment with its default evaluation settings (i.e., maximum episode length and success criteria) as defined in the original LIBERO benchmark.

C.2 Result

Reporting Protocol. All success rate values reported in this paper are the mean of 5 independent evaluation runs with different random seeds. We do not perform hyperparameter search for evaluation; all models are evaluated using their officially released or documented inference configurations.

D Analysis

D.1 Finding 1: Paraphrase Fragility Persists Across Architectures, Data Scales, and Fine-tuning Strategies

Fig. 17 presents per-model success rate heatmaps across all Object $\times$ Action paraphrase type combinations. While all seven models degrade under object paraphrasing, the degradation manifests in two distinct patterns. OpenVLA-OFT variants and VLA-Adapter exhibit a sharp cliff between object-preserved rows (None, Addition) and object-paraphrased rows (SP-contextual, SP-habitual) — the top two rows remain nearly uniformly green while the bottom two rows shift abruptly to red. This two-band pattern directly reflects the large preserved-vs-paraphrased gaps reported in Fig. 7: OpenVLA-OFT_{goal} (48.3 pp), OpenVLA-OFT_{mixed} (51.0 pp), and VLA-Adapter (37.1 pp) all show a clear visual boundary at the object paraphrasing threshold. In contrast, $\pi_{0.5}$, X-VLA, and Xiaomi-Robotics-0 display a more gradual degradation across both axes without a single sharp boundary, consistent with their comparatively smaller preserved-vs-paraphrased gaps (19.8–35.7 pp).

Despite these differences in degradation profile, the conclusion is shared: every model falls below 50% in the most challenging compound cells, confirming that paraphrase fragility is universal regardless of architecture.

D.2 Finding 2: Object Grounding Is the Primary Bottleneck

Alpha Sensitivity Analysis. Fig. 13 examines how the balance between the two PRIDE

Model	Pre-train	LIBERO Data	Fine-tune (Domain Adaptation)		Weights
	Pre-train Data		FT Method	FT Scope	
OpenVLA-OFT _{goal}	OXE (970k traj)	Goal only	LoRA (r=32)	All modules	Released
OpenVLA-OFT _{mixed}	OXE (970k traj)	All 4 suites	LoRA (r=32)	All modules	Released
$\pi_{0.5}$	Proprietary + open (10K+ hrs)	All 4 suites	Full	All modules	Released
$\pi_{0.5}$ (expert-only)	Proprietary + open (10K+ hrs)	All 4 suites	Full	Action Expert only	Ours
VLA-Adapter	No robotic pretrain	Goal only	LoRA (r=64)	All modules	Released
X-VLA	Droid + Robomind + Agibot (290k)	All 4 suites	LoRA (r=64)	All modules	Released
Xiaomi-Robotics-0	Open + in-house (~200M steps)	All 4 suites	Full	All modules	Released

Table 19: LIBERO fine-tuning configurations of the evaluated VLA models. All models are fine-tuned on LIBERO and evaluated on LIBERO-Para. “Released” denotes publicly available checkpoints; “Ours” denotes checkpoints fine-tuned by us following the original training protocol (see Fig. 15 for the training loss curve). Models trained on “All 4 suites” use the mixed configuration of LIBERO-Goal, LIBERO-Spatial, LIBERO-Object, and LIBERO-Long. Note that $\pi_{0.5}$ (expert-only) freezes the vision-language backbone and updates only the action expert.

	$\pi_{0.5}$	$\pi_{0.5}$ (expert-only)
VLM (img + llm)	Fine-tuned	Frozen
Action Expert	Fine-tuned	Fine-tuned
Trainable Params	~3.3B	~300M
Batch Size	256	256
Peak LR	5e-5	5e-5
Optimizer	AdamW (grad clip 1.0)	AdamW (grad clip 1.0)
EMA Decay	0.999	0.999
Warmup Steps	10k	10k
Training Steps	30k	30k
Action Horizon	10	10

Table 20: Training configurations for $\pi_{0.5}$ variants. The expert-only variant freezes the VLM and fine-tunes only the Action Expert.

components—keyword similarity (S_K) and structural similarity (S_T)—affects the overall robustness score. As α shifts toward 1.0 (keyword-centric), PRIDE scores decrease across all models, revealing that models generally succeed on samples where keywords are preserved and fail when keywords are paraphrased. Conversely, as α approaches 0.0 (structure-centric), scores rise uniformly, suggesting that structural variation alone is less disruptive than keyword replacement. This confirms that object-level keyword changes, rather than syntactic reformulations, are the dominant factor driving success rate degradation across current VLA architectures.

The per-model slopes in the right panel of Fig. 13 further reveal architecture-specific sensitivities. OpenVLA-OFT_{goal} and OpenVLA-OFT_{mixed} exhibit the steepest slopes (−17.3 and −18.7, respectively), consistent with the large success rate gaps between object-preserved and object-paraphrased conditions reported in Fig. 7 (48.3pp and 51.0pp, respectively). Their high keyword dependence indicates that these models rely heavily on exact object noun matching for task execution.

Interestingly, at $\alpha \geq 0.9$, X-VLA overtakes

OpenVLA-OFT_{mixed} in PRIDE score, indicating that despite lower structural robustness overall, X-VLA is more resilient to keyword-level variation. Similarly, $\pi_{0.5}$ (expert-only) closes the gap with VLA-Adapter at $\alpha = 1.0$, suggesting relatively stronger keyword robustness despite its lower absolute performance. These crossover patterns demonstrate the diagnostic utility of α -sweep: by adjusting the weighting, practitioners can identify which robustness dimension—keyword preservation or structural flexibility—a given model excels at, informing model selection for deployment environments where one linguistic dimension may be more prevalent than the other.

Action Indirectness. Fig. 14 breaks down success rate by action paraphrase type across all models. Lexical-level changes (Addition, SP-contextual, SP-habitual) cause moderate degradation (66–70%), while structural reorganizations (Coordination, Subordination) reduce success further to around 57–63%. The sharpest drops occur in the pragmatic category, where Question and Hint—forms that require pragmatic inference to recover the underlying imperative—bring success down to

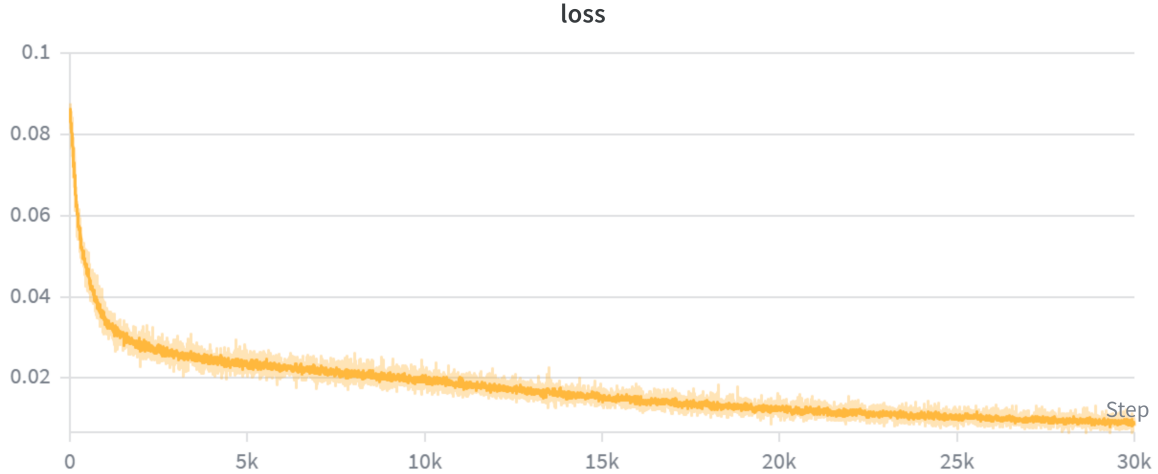


Figure 15: Training loss curve of $\pi_{0.5}$ (expert-only) fine-tuned on LIBERO. The model is trained for 30K steps, matching the original training configuration. The loss converges around 15K steps, indicating stable training completion.

~48%.

Notably, the overall action-axis degradation is milder than the object-axis degradation reported in Sec. 6.2. We attribute this to the constrained nature of tabletop manipulation: the action space is limited to a small set of motor primitives (pick, place, push, open, etc.), and each object typically affords only a narrow range of feasible actions (e.g., *stove* $\rightarrow$ *turn on*). This low action ambiguity allows models to converge on the correct primitive even under moderate linguistic variation. However, when the directive intent itself becomes opaque—as in questions or hints—models can no longer reliably extract the intended action, leading to the steep drop in the pragmatic category.

LIBERO-Goal Instructions. LIBERO-Goal instructions refer to each object by a single fixed reference throughout all tasks. As shown in Fig. 12, objects such as *stove*, *bowl*, and *rack* appear consistently under the same surface form, with no synonym or alternative reference used in any instruction. Because models are fine-tuned exclusively on these fixed references, they are never exposed to lexical variation in object references during training. This single-reference convention likely reinforces surface-level keyword matching and contributes to the sharp performance drops observed when object nouns are replaced with semantically equivalent alternatives in LIBERO-Para.

D.3 Finding 3: Failures Are Predominantly Planning-Level, Not Execution-Level

DTW-Based Trajectory Classification. We classify each failed episode as Near-GT (execution-level) or Far-GT (planning-level) based on its Dynamic Time Warping (DTW) distance to a pseudo ground-truth (GT) trajectory, as formalized in Algorithm 1.

Why DTW. Trajectory lengths vary across episodes—successful episodes may terminate early while failed episodes often run to the maximum step limit. Euclidean distance requires fixed-length inputs and cannot account for temporal misalignment between trajectories that follow similar spatial paths at different speeds. DTW handles both variable-length sequences and temporal warping, making it suitable for comparing manipulation trajectories. We use *fastdtw* (Salvador and Chan, 2007) with Euclidean distance as the local cost function, and normalize the resulting distance by sequence length to ensure comparability across episodes.

Resampling. To standardize input length for DTW, all trajectories are resampled to $K=50$ points via linear interpolation. This value was chosen as a practical trade-off between spatial resolution and computational cost across $\sim 143K$ total episodes ($4,092$ paraphrases $\times 5$ seeds $\times 7$ models).

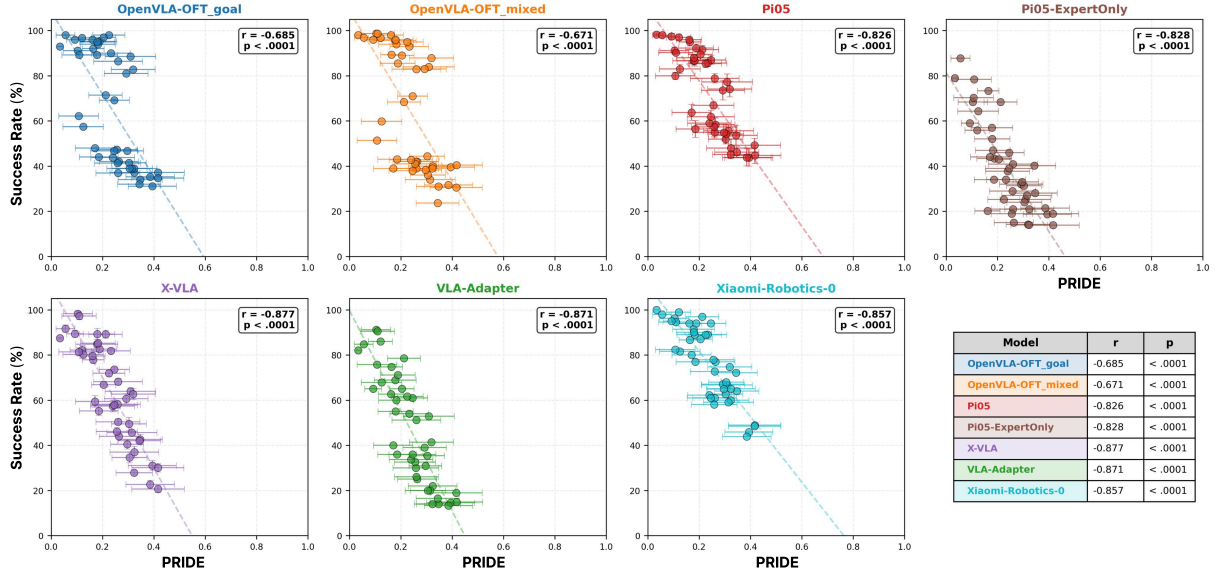


Figure 16: Correlation between PRIDE score (PD) and success rate (SR) for each VLA model on LIBERO-Para. Each point represents the mean SR of a paraphrase cell, with error bars indicating standard deviation. Colors are unified per model for visual clarity. All models exhibit statistically significant negative correlations ($p < .0001$), with Pearson r values ranging from -0.671 to -0.877 , validating that higher paraphrase distance consistently leads to lower task success. The summary table (bottom right) reports r and p for all models.

Near-GT % of Total					Far-GT % of Total				
Model	max	p99	p95	p90	Model	max	p99	p95	p90
OpenVLA-OFT _{goal}	1.6	1.4	0.4	0.3	OpenVLA-OFT _{goal}	33.7	33.9	34.9	35.0
OpenVLA-OFT _{mixed}	3.3	1.0	0.1	0.0	OpenVLA-OFT _{mixed}	33.0	35.3	36.2	36.3
$\pi_{0.5}$	2.4	1.0	0.6	0.3	$\pi_{0.5}$	26.2	27.6	28.0	28.3
$\pi_{0.5}$ (expert-only)	12.5	6.2	2.1	0.9	$\pi_{0.5}$ (expert-only)	48.4	54.7	58.8	60.0
VLA-Adapter	4.2	2.2	0.5	0.0	VLA-Adapter	49.5	51.5	53.2	53.7
X-VLA	5.2	2.9	0.8	0.4	X-VLA	32.7	35.0	37.1	37.5
Xiaomi-Robotics-0	1.8	0.3	0.1	0.0	Xiaomi-Robotics-0	22.2	23.7	23.9	24.0

Table 21: τ threshold ablation for trajectory-based failure classification. “max” denotes the most lenient threshold (widest Near-GT boundary); p99, p95, and p90 progressively tighten the criterion. Across all thresholds, Far-GT (planning-level) failures consistently dominate, confirming that the finding is robust to threshold selection.

EEF Position Only. From the 7-dimensional proprioceptive state $(x, y, z, r_x, r_y, r_z, g)$, we use only the first three dimensions corresponding to the end-effector (EEF) absolute position (x, y, z) . The remaining dimensions (orientation, gripper state) are excluded because spatial trajectory divergence is the most direct indicator of whether the model planned toward the correct target object—the core diagnostic question of this analysis.

Threshold Robustness. The threshold τ_t is set per-task as the maximum DTW distance among successful episodes (Algorithm 1, line 12), representing the most lenient Near-GT boundary. To verify that our findings are not sensitive to this choice, we repeat the classification with progressively stricter thresholds (p99, p95, p90 of success-

ful DTW distances). As shown in Tab. 21, Far-GT failures remain dominant across all thresholds—tightening τ shifts some Near-GT episodes to Far-GT but does not alter the overall conclusion. For example, even under the strictest criterion (p90), $\pi_{0.5}$ (expert-only) retains the highest Near-GT ratio among all models, consistent with the frozen-VLM interpretation discussed in Sec. 6.3.

GT Trajectory Consistency. Fig. 18 visualizes successful EEF trajectories for each LIBERO-Goal task. Within each task, successful trajectories converge to a narrow spatial corridor with low variance, validating the use of their mean as a pseudo GT. This consistency arises from the LIBERO-Goal training data, which contains a single fixed demonstration path per task with no route diversity.

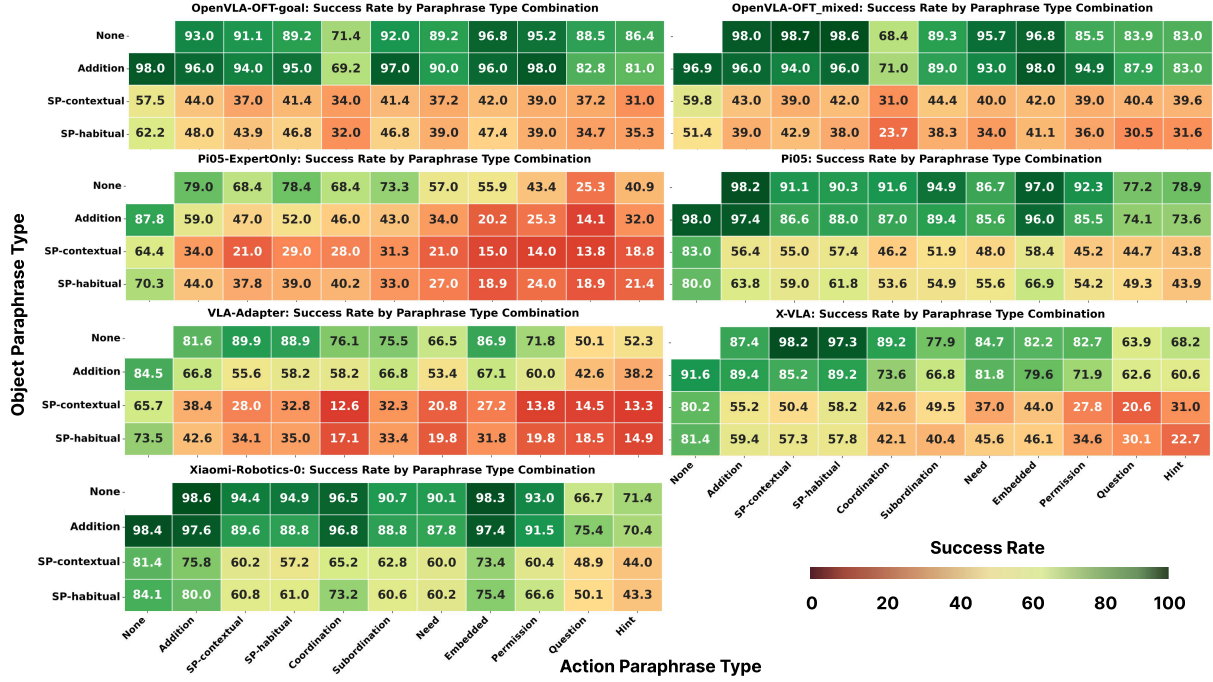


Figure 17: Per-model success rate heatmaps across all Object $\times$ Action paraphrase type combinations on LIBERO-Para. Rows represent object paraphrase types and columns represent action paraphrase types. Each cell reports the mean success rate over 5 seeds. The None row/column indicates the original (unparaphrased) instruction. All models show consistent degradation as paraphrase distance increases from the top-left (original) to the bottom-right (most distant) cells.

Per-Model Failure Decomposition. Fig. 21 provides a fine-grained view of the failure classification from Tab. 5, decomposed along the Object and Action axes for each model individually. Two observations are consistent across all models. First, Near-GT (execution-level) failures account for a small fraction in every category, confirming that the dominance of Far-GT failures reported in Sec. 6.3 is not an artifact of aggregation but holds at the per-type level. Second, Near-GT failures do not concentrate along any particular paraphrase axis or type—they are distributed roughly uniformly, suggesting that execution-level errors are not systematically triggered by specific linguistic properties.

The sole exception is $\pi_{0.5}$ (expert-only), which shows elevated Near-GT ratios across most categories. As discussed in Sec. 6.1, this model freezes the VLM during fine-tuning, preserving pretrained language understanding that enables partial task identification. However, the unadapted action expert lacks the precision to convert correct plans into successful executions, resulting in trajectories that track the GT path but ultimately fail.

These patterns reinforce the conclusion that paraphrase robustness improvements should target the instruction-to-task identification stage—where Far-

GT failures originate—rather than low-level motor control refinement.

E CALVIN-Para: Cross-Benchmark Validation

E.1 Setup

CALVIN Benchmark. CALVIN (Mees et al., 2021) is an open-source simulated benchmark for long-horizon, language-conditioned manipulation, in which a 7-DoF Franka Panda arm operates on a tabletop containing a sliding door, a drawer, colored blocks, an LED, and a light bulb. It comprises four environments (A–D) that differ in desk texture and object layout, and defines 34 manipulation tasks with natural-language instructions. In the standard protocol, policies are evaluated on 1,000 chains of five consecutive language-conditioned sub-tasks, where the initial state is instantiated for the first sub-task of each chain and success on later sub-tasks depends on earlier ones.

Benchmark Construction. We construct CALVIN-Para by applying the two-axis paraphrase design of LIBERO-Para (Sec. 3) to CALVIN (Mees et al., 2021) under the standard ABCD $\rightarrow$ D split. From the 34 original instructions, we select 15

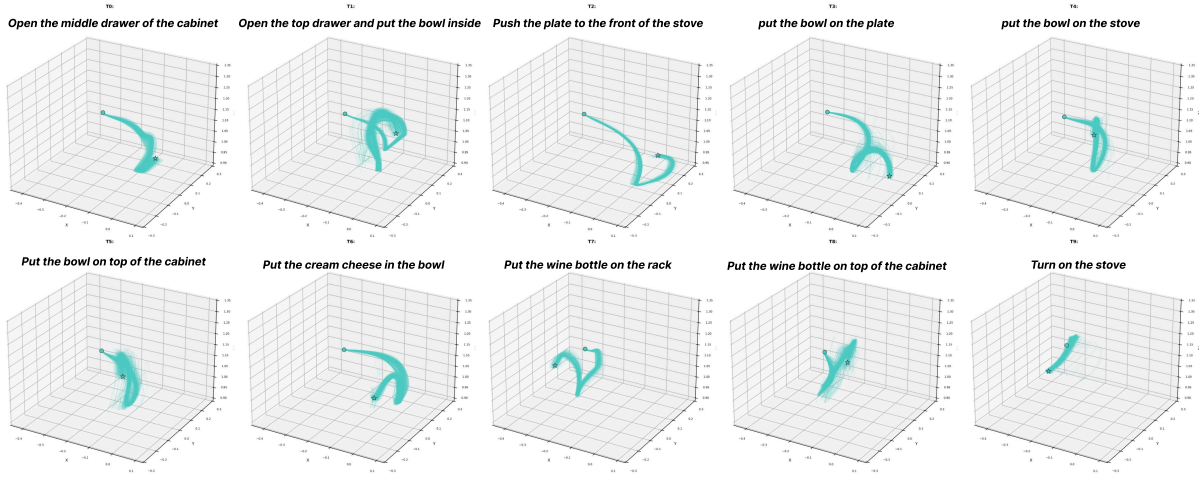


Figure 18: Successful EEF trajectories of Xiaomi-Robotics-0 on LIBERO-Para, grouped by LIBERO-Goal task index (T0–T9). Within each task, successful trajectories converge to a narrow corridor with low spatial variance, indicating that manipulation strategies are largely invariant to paraphrase variation. We observe consistent patterns across all evaluated models; a single model is shown for visual clarity. This consistency motivates the use of the mean successful trajectory as a pseudo ground-truth (GT) in Algorithm 1.

	RoboFlamingo	FLOWER
Variant	MPT-IFT-3B	—
Split	ABCD→D	ABCD→D
Code	github.com/RoboFlamingo/RoboFlamingo	github.com/intuitive-robots/flower_vla_calvin
Weights	hf.co/robovlms/RoboFlamingo	hf.co/mbreuss/flower_calvin_abcd

Table 22: Official pretrained checkpoints used for CALVIN-Para evaluation.

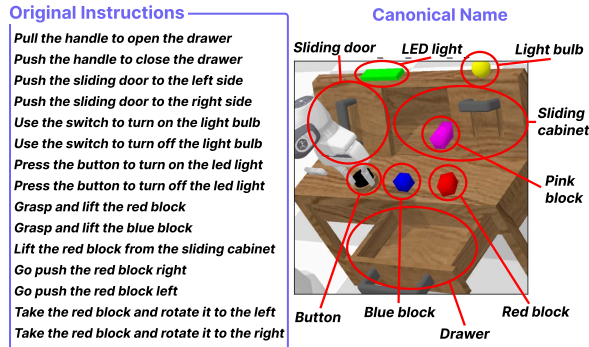


Figure 19: Base instructions used in CALVIN-Para (left) and corresponding scene with canonical object names (right). As in LIBERO-Goal (Fig. 12), each object is referred to by a single unique keyword throughout all instructions (e.g., *drawer*, *sliding door*, *red block*), with no lexical variation across tasks.

base tasks in two stages. First, we exclude 14 tasks distinguished solely by object color (e.g., lifting the red, blue, or pink block), since color terms admit few natural synonyms and thus cannot support meaningful object-axis paraphrasing; tasks in which a colored object is the sole referent are retained. Second, we exclude 5 tasks (e.g.,

place_in_drawer, *stack_block*) that never appear in the first slot of CALVIN’s 5-task evaluation chains: their initial states require preceding manipulation and cannot be instantiated without altering CALVIN’s original initial-state distribution, which would break the canonical success-rate comparison. Each of the 43 variation cells contains 3 paraphrases per base task (45 instructions per cell), yielding 1,935 paraphrased instructions in total. Fig. 19 shows the 15 base instructions and the corresponding scene.

Evaluation Protocol. To isolate paraphrase effects, we convert CALVIN’s 5-step task chains into a single-task setting. Unlike LIBERO-Goal, initial scenes vary across episodes; the language instruction nevertheless serves as the task cue, so the effect of paraphrasing is measured cleanly. We adopt the ABCD→D split so that the evaluation environment lies within the training distribution: tasks are seen under canonical instructions while only the paraphrases are unseen, matching the seen-task, unseen-instruction setting of LIBERO-Para.

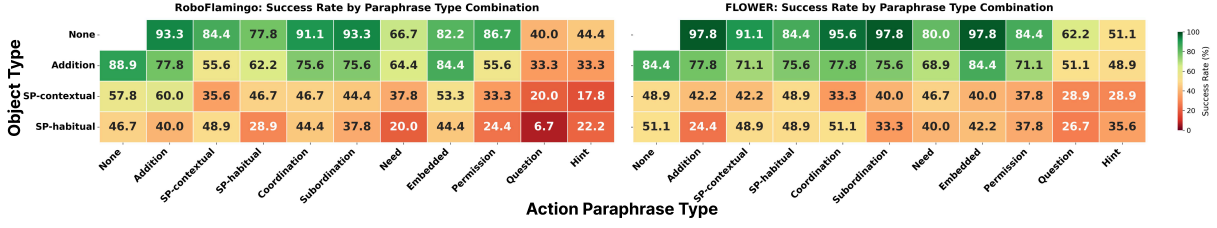


Figure 20: Per-model success rate heatmaps across all Object $\times$ Action paraphrase type combinations on CALVIN-Para (left: RoboFlamingo, right: FLOWER). Rows represent object paraphrase types and columns represent action paraphrase types. Each cell reports the mean success rate over 5 seeds. The None row/column indicates the original (unparaphrased) instruction. As on LIBERO-Para (Fig. 17), both models show consistent degradation as paraphrase distance increases from the top-left (original) to the bottom-right (most distant) cells.

Method	CALVIN SR	CALVIN-Para SR (Drop)
RoboFlamingo	92.3	53.1 (-39.2)
FLOWER	99.7	58.3 (-41.4)

Table 23: Success rate (SR) comparison between CALVIN and CALVIN-Para. Drop denotes the absolute decrease in success rate.

Method	SR	PRIDE	Overestimation (%)
RoboFlamingo	53.1	43.5	18.1
FLOWER	58.3	48.3	17.2

Table 24: SR and PRIDE scores on CALVIN-Para, sorted by overestimation. Overestimation is computed as $(SR - PRIDE) / SR$, indicating how much uniform success rate overstates a model’s paraphrase robustness. PRIDE is computed under the same formulation as Tab. 3 (Eq. 4).

Models. We evaluate two CALVIN-native VLAs not used in our LIBERO experiments, RoboFlamingo (Li et al., 2024) and FLOWER (Reuss et al., 2025), using official ABCD $\rightarrow$ D pretrained weights without modification (Tab. 22). All inference is run on NVIDIA L40S GPUs, and all results are averaged over 5 seeds.

E.2 Results

Success Rate Comparison. As shown in Tab. 23, despite near-saturated performance of 92.3% and 99.7% on canonical CALVIN instructions, RoboFlamingo and FLOWER drop by 39.2 and 41.4 pp under paraphrasing, within the 22.8–51.9 pp range observed on LIBERO-Para in Tab. 2. At cell level, Fig. 20 shows the same overall tendency as on LIBERO-Para: success rates degrade monotonically along both axes, from the top-left (original) toward the bottom-right (most distant) cells.

Method	Obj. preserved	Obj. paraphrased (Δ)
RoboFlamingo	69.8	37.2 (-32.6)
FLOWER	77.6	39.9 (-37.7)

Table 25: Mean success rate for object-preserved (None, Addition) versus object-paraphrased (SP-contextual, SP-habitual) instructions on CALVIN-Para, following Fig. 7. Δ denotes the absolute decrease in success rate.

Model	Success Rate	Failure		
		Near-GT	Far-GT	Far-GT(%)
FLOWER	58.3	1.2	40.5	97.1
RoboFlamingo	53.1	3.1	43.8	93.4

Table 26: Failure classification on CALVIN-Para (by Far-GT (%)). Near-GT: execution-level failure near GT trajectory. Far-GT: planning-level failure from GT. As in LIBERO-Para (Tab. 5), the vast majority of failures are planning-level.

PRIDE Reveals Hidden Severity. Tab. 24 shows that uniform SR overstates robustness by 18.1% and 17.2% relative to PRIDE, comparable to the most-overestimated models on LIBERO-Para in Tab. 3. PRIDE thus captures the hidden severity of paraphrase failures across benchmarks.

Finding Reproduction. All three findings from Sec. 6 reproduce on CALVIN-Para. First, the two models instantiate architecture designs absent from our LIBERO experiments: RoboFlamingo couples a Flamingo-style VLM (Alayrac et al., 2022), in which language tokens attend to visual features through gated cross-attention, with a separate LSTM (Hochreiter and Schmidhuber, 1997) policy head for temporal modeling, while FLOWER fuses a compact Florence-2 encoder (Xiao et al., 2024) with a rectified-flow (Liu et al., 2022) DiT (Peebles and Xie, 2023) action expert at an intermediate VLM layer. Both nonetheless exhibit the same fragility, extending Finding 1 in Sec. 6.1: para-

phrase fragility persists regardless of how vision, language, and action generation are coupled. Second, as shown in Tab. 25, replacing canonical object names degrades SR by 32.6 and 37.7 pp, mirroring the 19.8–51.0 pp drops in Fig. 7 and reproducing Finding 2 in Sec. 6.2: models remain anchored to the single canonical object word seen during training. Third, Tab. 26 shows that 93.4% and 97.1% of failures are Far-GT, consistent with the 79.5–95.5% range in Tab. 5 and reproducing Finding 3 in Sec. 6.3: failures are predominantly planning-level rather than execution-level.

F AI Assistants

During the course of this work, we used Google’s Gemini 2.5 Pro (<https://gemini.google.com/>) (Comanici et al., 2025) for generating paraphrase candidates in the LIBERO-Para benchmark construction. All generated paraphrases were manually reviewed and filtered by the authors. Additionally, we used AI assistants including OpenAI’s ChatGPT (<https://chatgpt.com/>) (OpenAI, 2023) and Anthropic’s Claude (<https://claude.ai/>) (Anthropic, 2025) to proofread and improve the clarity of our writing. We affirm that these tools served solely as assistive aids and did not contribute to core research ideas, experimental design, analysis, or interpretation of results. The final scientific content and all claims made in this paper are the sole responsibility of the authors.

Near-GT / Far-GT Failure Analysis per Model

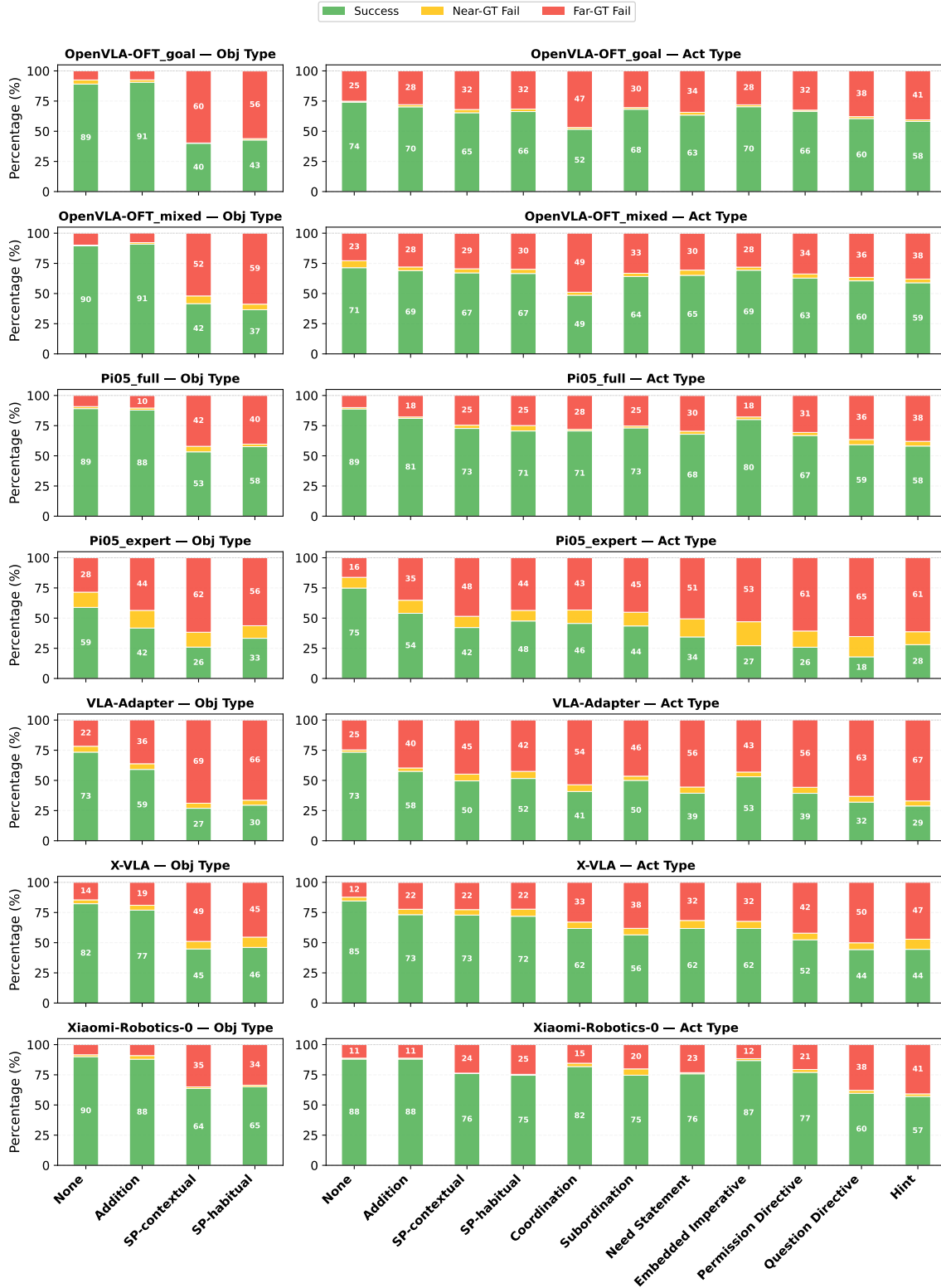


Figure 21: Near-GT / Far-GT failure breakdown per model, decomposed by Object axis (left) and Action axis (right). Each bar shows the proportion of Success (green), Near-GT failure (yellow, execution-level), and Far-GT failure (red, planning-level) episodes. The threshold τ is set to the maximum DTW distance among successful episodes per task. Across all models and paraphrase types, Far-GT failures consistently dominate, with no concentration of Near-GT failures along any specific axis. The exception is $\pi_{0.5}$ (expert-only), which exhibits a higher Near-GT ratio due to its frozen VLM preserving partial task identification while the unadapted action expert fails at execution.

Algorithm 1 Trajectory-based failure classification for Sec. 6.3 A pseudo ground-truth trajectory (GT) is constructed from successful episodes of each model on LIBERO-Para. Failed episodes are classified as Near-GT (execution-level) or Far-GT (planning-level) based on DTW distance, with the threshold τ_t set to the maximum distance among successes.

Require: Set of episodes $\mathcal{E} = \{e_1, \dots, e_N\}$ for original LIBERO-Goal task index $t \in \{0, 1, \dots, 9\}$, each with trajectory $\tau_i \in \mathbb{R}^{T_i \times 3}$ and outcome $s_i \in \{0, 1\}$; resampling size $K=50$

Ensure: Classification of each failed episode as NEAR-GT or FAR-GT

// Step 1: Partition episodes

- 1: $\mathcal{S}_t \leftarrow \{e_i \in \mathcal{E} \mid s_i = 1\}$ ▷ successes
- 2: $\mathcal{F}_t \leftarrow \{e_i \in \mathcal{E} \mid s_i = 0\}$ ▷ failures

// Step 2: Construct pseudo-GT trajectory

- 3: **for** each $e_i \in \mathcal{S}_t$ **do**
- 4: $\hat{\tau}_i \leftarrow \text{RESAMPLE}(\tau_i[:, : 3], K)$ ▷ first 3 dims of proprio: EEF absolute position (x,y,z)
- 5: **end for**
- 6: $\tau^{\text{GT}} \leftarrow \frac{1}{|\mathcal{S}_t|} \sum_{e_i \in \mathcal{S}_t} \hat{\tau}_i$

// Step 3: Compute DTW distances

- 7: $L_{\max} \leftarrow \max_{e_j \in \mathcal{S}_t} T_j$
- 8: **for** each $e_i \in \mathcal{F}_t$ **do**
- 9: $\tau'_i \leftarrow \text{RESAMPLE}(\tau_i[:, L_{\max}, : 3], K)$
- 10: $d_i \leftarrow \text{DTW}(\tau'_i, \tau^{\text{GT}})$ ▷ DTW: Dynamic Time Warping
- 11: **end for**

// Step 4: Threshold

- 12: $\tau_t \leftarrow \max_{e_i \in \mathcal{S}_t} d_i$

// Step 5: Classify failures

- 13: **for** each $e_i \in \mathcal{F}_t$ **do**
 - 14: **if** $d_i \leq \tau_t$ **then**
 - 15: $\text{LABEL}(e_i) \leftarrow \text{NEAR-GT}$ ▷ execution-level
 - 16: **else**
 - 17: $\text{LABEL}(e_i) \leftarrow \text{FAR-GT}$ ▷ planning-level
 - 18: **end if**
 - 19: **end for**
 - 20: **return** $\{\text{LABEL}(e_i)\}_{e_i \in \mathcal{F}_t}$
-

Generator Prompt

Paraphrase the given robot manipulation instruction while preserving semantic meaning.

Rules:

- Preserve plurality (singular/plural must match original)
- Do not add visual attributes (color, size, shape, material)
- Do not add spatial attributes (position, location, direction)
- Modify only what is specified in the task guidelines below

Scope:

- Object tasks: modify only object nouns, preserve verbs and structure
- Action tasks: modify only action elements, preserve object nouns

Output: One paraphrase per line. No explanations or alternatives.

Figure 22: Common prompt template for the Paraphrase Generator (LLM), shared across all paraphrase types.

Verifier Prompt

Criteria:

- Task compliance: Required transformations applied, prohibited changes avoided.
 - Object tasks (obj_*): only object nouns changed, verbs/structure intact
 - Action tasks (act_*): only action elements changed, object nouns intact
- Semantic preservation: Core action and target objects unchanged.
- Naturalness: Grammatically correct, natural phrasing.
- Format: Single instruction only, no meta-commentary or explanations.
- Lexical clarity: Avoid confusion with environment objects (stove, bowl, plate, rack, wine bottle, cream cheese, cabinet).
 - Acceptable: “cupboard” (distinct from cabinet)
 - Reject: “dish” (confusable with plate/bowl)

Output: Accepted paraphrases only, one per line.

Figure 23: Common prompt template for the Paraphrase Verifier (LLM), used to filter generated paraphrases.

Merge Generator Prompt

Merge object paraphrases and action paraphrases into combined variants.

Input:

- Original instruction
- Object paraphrase examples (nouns changed)
- Action paraphrase examples (verbs/structure changed)

Task: Create paraphrases with both object and action modifications applied.

Process:

1. Identify object substitution patterns (e.g., “drawer” → “compartment”)
2. Identify action modification patterns (e.g., “pick” → “grab”)
3. Apply both transformations coherently

Example:

- Original: “*pick the bowl and place on the stove*”
- Object variant: “*pick the container and place on the cooktop*”
- Action variant: “*grab the bowl and put on the stove*”
- Merged: “*grab the container and put on the cooktop*”

Output: 5–10 merged paraphrases, one per line. No numbering or explanations.

Figure 24: Prompt template for combining validated Object and Action paraphrases into merged variants.

Merge Verifier Prompt

Evaluate merged paraphrases that combine object and action changes.

Criteria:

- Completeness: Both object nouns AND action elements must differ from original. Reject if only one type changed.
- Pattern consistency: Changes follow the provided examples.
- Semantic preservation: Task intent and outcome unchanged.
- Naturalness: Grammatically correct, coherent combination.
- Format: Single instruction, no meta-commentary.
- Lexical clarity: No confusion with environment objects.

Output: Accepted paraphrases only, one per line.

Figure 25: Prompt template for verifying merged paraphrases before final inclusion in the dataset.

Object-Lexical Types Prompts

obj_lexical_same_polarity_habitual

Replace object nouns with general synonyms.

EPT: Same-polarity substitution (habitual)

Guidelines:

- Use commonly accepted synonyms (bowl → cup, cabinet → cupboard)
- Preserve grammatical form
- Keep verbs and structure unchanged

Examples:

- *“Pick the bowl and place on the stove” → “Pick the cup and place on the stove.”*
- *“Open the middle layer of drawer” → “Open the middle layer of compartment.”*

obj_lexical_same_polarity_contextual

Replace object nouns with contextually appropriate alternatives.

EPT: Same-polarity substitution (contextual)

Guidelines:

- Use contextually similar items (bowl → container, stove → cooking surface)
- Maintain semantic appropriateness for manipulation tasks
- Preserve structure

Examples:

- *“Pick the bowl and place on the stove” → “Pick the container and place on the cooking surface.”*
- *“Open the middle layer of drawer” → “Open the middle tier of drawer.”*

obj_lexical_addition

Add functional descriptors from object names.

EPT: Addition

Guidelines:

- Add functional/categorical adjectives (soup bowl, kitchen cabinet)
- Exclude visual adjectives (color, size, material)
- Exclude spatial adjectives (top, left, big)
- Preserve plurality

Examples:

- *“Pick the bowl and place on the stove” → “Pick the mixing bowl and place on the kitchen stove.”*
- *“Open the middle layer of drawer” → “Open the middle layer of storage drawer.”*

Figure 26: Type-specific generation guidelines for Object-Lexical paraphrases. These guidelines are provided to the Generator for paraphrase generation and also appended to the Verifier to assess whether the generated output conforms to the intended variation type.

Action-Lexical Types Prompts

act_lexical_same_polarity_habitual

Replace action verbs with general synonyms.

EPT: Same-polarity substitution (habitual)

Guidelines:

- Use common verb synonyms (pick → grab, place → put)
- Preserve structure and arguments
- Keep object nouns unchanged

Examples:

- “*Pick the bowl and place on the stove*” → “*Grab the bowl and put on the stove.*”
- “*Open the middle layer of drawer*” → “*Pull the middle layer of drawer.*”

act_lexical_same_polarity_contextual

Replace action verbs with contextually appropriate alternatives.

EPT: Same-polarity substitution (contextual)

Guidelines:

- Use context-appropriate alternatives (pick → grasp, place → position)
- Preserve core action meaning
- Keep object nouns unchanged

Examples:

- “*Pick the bowl and place on the stove*” → “*Grasp the bowl and position on the stove.*”
- “*Open the middle layer of drawer*” → “*Access the middle layer of drawer.*”

act_lexical_addition_deletion

Add or remove manner adverbs.

EPT: Addition/Deletion

Guidelines:

- Add single-word adverbs (carefully, gently, slowly)
- For phrasal additions, use act_structural_ellipsis instead
- Keep verb and structure unchanged

Examples:

- “*Pick the bowl and place on the stove*” → “*Carefully pick the bowl and place on the stove.*”
- “*Open the middle layer of drawer*” → “*Gently open the middle layer of drawer.*”

Figure 27: Type-specific generation guidelines for Action-Lexical paraphrases. These guidelines are provided to the Generator for paraphrase generation and also appended to the Verifier to assess whether the generated output conforms to the intended variation type.

Action-Structural Types Prompts

act_structural_coordination

Modify coordination structure.

EPT: Coordination changes (syntax-based)

Guidelines:

- Split into separate sentences
- Add explicit ordering (First... then...)
- Combine with coordination
- Preserve all information

Examples:

- “*Pick the bowl and place on the stove*” → “*Pick the bowl. Place it on the stove.*”
- “*Open the middle layer of drawer*” → “*Locate the drawer, then open the middle layer.*”

act_structural_subordination

Modify subordination structure.

EPT: Subordination changes (syntax-based)

Guidelines:

- Convert coordination to subordination
- Use temporal subordinators (after, once, when)
- Use purpose subordinators (so that, in order to)
- Preserve semantic relations

Examples:

- “*Pick the bowl and place on the stove*” → “*After picking the bowl, place it on the stove.*”
- “*Open the middle layer of drawer*” → “*Open the middle layer of drawer in order to access the contents.*”

Figure 28: Type-specific generation guidelines for Action-Structural paraphrases. These guidelines are provided to the Generator for paraphrase generation and also appended to the Verifier to assess whether the generated output conforms to the intended variation type.

Action-Pragmatic Types Prompts (1/2)

act_pragmatical_need_statement

Express as speaker's need.

Ervin-Tripp Type 1: Need Statements

Guidelines:

- Convert to first-person need (I need, I want, I require)
- Focus on desired outcome rather than process
- Preserve core action and objects

Examples:

- *"Pick the bowl and place on the stove" → "I need the bowl placed on the stove."*
- *"Open the middle layer of drawer" → "I want the middle layer of drawer open."*

act_pragmatical_embedded_imperative

Embed in question frame with modals.

Ervin-Tripp Type 3: Embedded Imperatives

Guidelines:

- Use modal questions (Could you, Would you, Can you)
- Keep agent and action explicit but softened

Examples:

- *"Pick the bowl and place on the stove" → "Could you pick the bowl and place it on the stove?"*
- *"Open the middle layer of drawer" → "Can you open the middle layer of drawer?"*

act_pragmatical_permission_directive

Frame as permission request.

Ervin-Tripp Type 4: Permission Directives

Guidelines:

- Request permission or access (May I have, Can I have access to)
- Focus on speaker's access requiring hearer's action

Examples:

- *"Pick the bowl and place on the stove" → "May I have the bowl placed on the stove?"*
- *"Open the middle layer of drawer" → "Can I have access to the middle layer of drawer?"*

Figure 29: Type-specific generation guidelines for Action-Pragmatic paraphrases (a). These guidelines are provided to the Generator for paraphrase generation and also appended to the Verifier to assess whether the generated output conforms to the intended variation type.

Action-Pragmatic Types Prompts (2/2)

act_pragmatical_question_directive

Pose information question implying action.

Ervin-Tripp Type 5: Question Directives

Guidelines:

- Ask contextually relevant question
- Do not explicitly specify desired act
- Action inferred from context

Examples:

- *“Pick the bowl and place on the stove”* → *“Is the bowl still sitting on the counter?”*
- *“Open the middle layer of drawer”* → *“Do we know the contents of the middle layer of drawer?”*

act_pragmatical_hint

Make statement implying action through inference.

Ervin-Tripp Type 6: Hints

Guidelines:

- Use general statement, not surface directive
- State condition or desired state
- Rely on situational inference
- Do not mention action directly

Examples:

- *“Pick the bowl and place on the stove”* → *“The stove surface is now clear for the bowl.”*
- *“Open the middle layer of drawer”* → *“The middle layer of drawer is still closed.”*

Figure 30: Type-specific generation guidelines for Action-Pragmatic paraphrases (b), continued. These guidelines are provided to the Generator for paraphrase generation and also appended to the Verifier to assess whether the generated output conforms to the intended variation type.