WHAT MAKES GOOD IN-CONTEXT EXAMPLES FOR VISION-LANGUAGE-ACTION MODELS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action models (VLAs) have demonstrated strong capabilities across a wide range of physical manipulation tasks. To enable more versatile generalization, recent studies have sought to improve their In-Context Learning (ICL) ability, *i.e.*, to perform new physical tasks by conditioning on a few related demonstrations at inference time. However, what information in these demonstrations actually drives ICL performance remains poorly understood. Prior work largely focuses on demonstrating ICL gains, leaving the roles of different context components underexplored. In this paper, we systematically investigate what makes in-context demonstrations effective for VLAs. We find that the *action tokens* in the demonstrations play the most important role, whereas removing images and language results in only a marginal decrease in performance. Our analysis also reveals a surprising fact: the pairing between observations and actions is largely unnecessary. Given a selected context, the *correct mapping* between images and action tokens is actually *not* required to perform ICL. Shuffling the ground-truth action tokens among demonstrations in the context window has little effect on the performance. We further find that ICL becomes more beneficial as tasks require greater generalization, while increasing the number of demonstrations can eventually hurt performance. Together, these findings provide practical insights into effective context construction and raise fundamental questions about how current VLAs use demonstrations for in-context learning.

1 INTRODUCTION

Vision-Language-Action (VLA) models are rapidly advancing robotic learning toward general-purpose policies. By jointly modeling visual observations, language instructions, and robot actions, recent VLAs (Intelligence et al., 2025; Kim et al., 2025b; Wu et al., 2026b; NVIDIA et al., 2025; Yu et al., 2026) have demonstrated strong performance across diverse manipulation tasks, environments, and embodiments. Beyond learning broad capabilities during training, an increasingly important question is whether these models can also *adapt at inference time* to tasks unseen at training time. Motivated by the success of In-Context Learning (ICL) in large language models (Brown et al., 2020), recent work has begun to equip VLAs with a similar capability: given a small number of related demonstrations as context, the model can improve its behavior on a target task without task-specific gradient updates (Sridhar et al., 2025a). Such a capability offers an appealing path toward robots that can quickly adapt to new tasks from only a few demonstrations.

However, despite growing evidence that contextual demonstrations can improve VLA performance, we still understand little about *what makes those demonstrations useful*. Existing studies (Sridhar et al., 2025a) primarily train VLAs with ICL abilities, add ICL abilities to pre-trained VLAs, and measure whether providing demonstrations improves downstream task performance. In comparison, the information contained *within* a demonstration has received much less attention. A robotic demonstration is a rich multimodal sequence: it contains visual observations, language, and action tokens at specific timestamps. It is still unclear *which of these components actually drives the observed ICL gains and what makes a good in-context example for VLAs*. Investigating this question is important both for understanding current VLAs' ICL capabilities and for designing more effective context-aware policies. During ICL, a VLA observes one or more demonstrations showing how states, instructions, and actions relate, and uses them to determine how to respond to a query input. Under this intuitive interpretation, visual observations, action sequences, and their correspondence should

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

all provide meaningful task information. However, improved performance with demonstrations does not necessarily imply that the model uses all of this information. A VLA may instead rely predominantly on a single modality, exploit coarse statistics of the demonstrated actions, or benefit from contextual properties that do not require recovering the demonstrated state–action relationship. In this work, we systematically examine the information contained in in-context demonstrations using two representative VLA models, $\pi_{0.5}$ (Intelligence et al., 2025) and InternVLA (Ma et al., 2026), across four robotic manipulation benchmarks: LIBERO (Liu et al., 2023), LIBERO-Plus (Fei et al., 2026), LIBERO-PRO (Zhou et al., 2025), and LIBERO-Mem (Chung et al., 2026) to mainly answer four research questions: *1. Which modalities in a demonstration matter most? 2. Does the correct correspondence between observations and actions matter? 3. How does the benefit of ICL vary with the generalization difficulty of the task? 4. Does providing more demonstration information necessarily improve performance?* We study these questions through controlled interventions on demonstration content, cross-modal correspondence, task generalization difficulty, and trajectory length across multiple VLA models and robotic manipulation settings.

Our experiments reveal several surprising properties of VLA in-context learning. First, we find that *action tokens are by far the most important component of the demonstrations*: removing actions causes the largest performance degradation, whereas removing images or language leads to only marginal decreases. Second, once a set of demonstrations has been selected, we find that the *correct mapping between visual observations and action sequences is not necessary for obtaining the ICL benefit*. Shuffling the ground-truth action sequences among demonstrations, *i.e.*, breaking their original image–action correspondence, has little effect on downstream performance. Third, we find that *the benefit of ICL becomes more pronounced as tasks require greater generalization*, with larger performance gains observed under more challenging generalization settings. Finally, we find that *more context is not always better*: the length of the provided demonstrations meaningfully affects performance, and longer demonstrations can even reduce the model's ability to solve the target task.

Together, these results challenge the straightforward interpretation as learning a coherent state–action mapping from a small number of multimodal examples. Instead, they suggest that current models may derive a substantial fraction of their ICL benefit from action-side information and other properties of the contextual sequence that do not require preserving the full multimodal structure of the demonstrations. Beyond providing practical guidance for constructing effective in-context examples, these findings expose important open questions about what current VLAs learn from context and what forms of inference-time adaptation future models should aim to support. To summarize, our main contributions are as follows:

- We show that *action tokens in the demonstrations contribute the most for ICL*: removing action tokens substantially degrades performance, while removing visual observations or language has only a marginal effect.
- We uncover the surprising result that *correct image–action correspondence is largely unnecessary* for the observed ICL gains: shuffling ground-truth action sequences across demonstrations produces little performance degradation.
- We demonstrate that *the amount of contextual information has a non-monotonic effect on VLAs*, with longer demonstrations not necessarily improving performance and, in some cases, reducing it.
- These findings provide new evidence about what makes good in-context examples for VLAs and motivate a more careful characterization of how current models use demonstration context.

2 Related Work

2.1 Generalist Vision–Language–Action Models

Recent vision–language–action (VLA) models have made substantial progress toward general-purpose robotic manipulation. Models such as OpenVLA-OFT (Kim et al., 2025a), RynnVLA (Cen et al., 2025), X-VLA (Zheng et al., 2026), and LingBot-VLA (Wu et al., 2026a) leverage large-scale robot data and pretrained multimodal representations to support language-conditioned behavior across diverse tasks, scenes, and embodiments. Among these models, $\pi_{0.5}$ demonstrates broad generalization across environments (Intelligence et al., 2025), while its publicly available LIBERO checkpoint and established evaluation pipeline facilitate further research on policy adaptation. InternVLA-A1.5

explores a complementary architectural direction by integrating vision–language understanding, latent foresight, and action generation: foresight tokens learn task-relevant future information under supervision from a pretrained video model (Ma et al., 2026). Together, these models illustrate complementary advances in generalist VLAs, from broader cross-environment competence to the integration of predictive representations into action generation.

2.2 In-Context Learning for Robot Policies

In-context learning enables robot policies to adapt from demonstrations supplied at inference time without task-specific parameter updates. Building on transformers that learn from interaction histories (Laskin et al., 2022) and manipulation policies conditioned on demonstrations (Fu et al., 2024; Vosylius & Johns, 2025), recent work has explored diverse forms of context for robot adaptation. These include retrieved experience in generalist policies (Sridhar et al., 2025b), video demonstrations in S1 (AI, 2026), sensorimotor examples in GEN-1.5 (Team, 2026), and execution-grounded trajectories in RoboICL (Liu et al., 2026). Among these approaches, RICL demonstrates how in-context adaptability can be added to a pretrained VLA (Sridhar et al., 2025a). VLAs are originally trained to predict actions from current observations and instructions, without additional demonstrations as context. Incorporating demonstrations directly into the input leads to model collapse. RICL addresses this limitation by post-training the policy with retrieved demonstrations, then retrieving relevant examples at inference time without task-specific parameter updates. Because it builds on an existing VLA rather than requiring a policy designed for in-context learning from the outset, RICL provides a practical approach to equipping pretrained robot policies with a demonstration-based context interface. Although these studies show that contextual examples can improve adaptation, the respective contributions of the context remain insufficiently understood. Our work examines the properties of context to better understand what drives the observed improvements.

2.3 Evaluation of VLA Generalization and Adaptation

Existing VLA evaluations primarily assess how well policies generalize across tasks and environments. LIBERO studies knowledge transfer across language-conditioned manipulation tasks (Liu et al., 2023), while LIBERO-Plus and LIBERO-PRO evaluate robustness under controlled distribution shifts (Fei et al., 2026; Zhou et al., 2025). In particular, LIBERO-Plus extends the LIBERO ecosystem with a large collection of task variants spanning seven perturbation dimensions and predefined difficulty levels, while retaining compatible robot, observation, and action interfaces (Fei et al., 2026). This structure supports controlled comparisons with baseline policies across different types and degrees of distribution shift. SimplerEnv, VLABench, and VLA-Arena further broaden evaluation across environments and task variations (Li et al., 2024; Zhang et al., 2024; 2026). Focusing more directly on in-context adaptation, LMAct examines how interactive performance changes as more demonstrations are provided (Ruoss et al., 2025). These benchmarks emphasize task-level outcomes, such as success under distribution shifts or performance gains from additional demonstrations. Our evaluation instead isolates the effects of individual contextual signals, examining how demonstration content and structure influence adaptation across perturbation types and difficulty levels.

3 Preliminaries

3.1 In-Context Learning for VLAs

In-context learning (ICL) allows a model to condition its predictions on examples provided at inference time without updating its parameters. For VLAs, these examples demonstrate robot behavior and provide additional information for action prediction. At each action-prediction step t, the policy receives a current query $q_t = (v_t, s_t, \ell)$, where v_t denotes the current visual observations, s_t the robot's proprioceptive state, and ℓ the task instruction. Without demonstration context, the policy predicts an action chunk as $\hat{A}_t = \pi_\Theta(q_t)$, where $\hat{A}_t = (\hat{a}_t, \ldots, \hat{a}_{t+H-1})$ contains H predicted actions and Θ denotes the policy parameters. With demonstration context, the policy additionally receives an ordered collection of K examples, denoted by E_t, to generate the action tokens:

$$\hat{A}_t = \pi_\Theta(q_t, E_t). \tag{1}$$

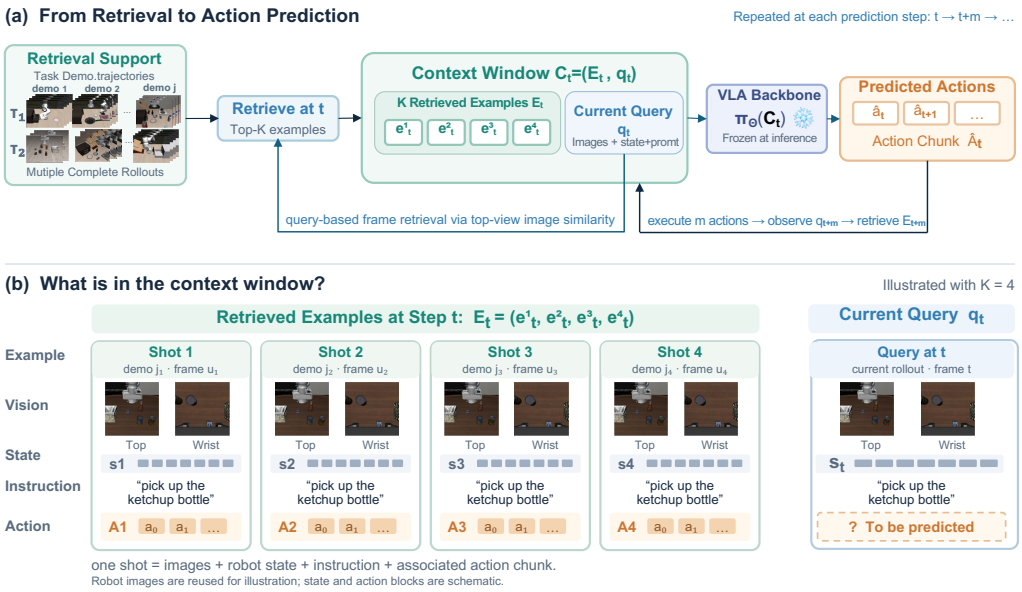


Figure 1: Context window for VLA.The current query q_t is used to retrieve K demonstration examples from the retrieval bank. These examples form the demonstration context E_t. Together with q_t, they constitute the complete context window C_t, which the policy uses to predict the action $\hat{A}_t$.

As illustrated in Fig. 1, q_t is used to retrieve demonstration examples from the retrieval support. The retrieved examples E_t and the current query q_t jointly form the context window supplied to the policy. The policy can be fine-tuned beforehand to better use demonstration context (Sridhar et al., 2025a), but all parameters Θ remain fixed during ICL evaluation. To better examine the contribution of each component in the context, we first analyze the structure and information of each example and how these examples are combined with the current query.

3.2 From Individual Shots to the Context Window

For a given query task, we consider a task-specific retrieval support containing demonstration trajectories for that task. Let j index a demonstration trajectory within this support, and let u index a frame within that trajectory. In our setting, one *shot* is a *frame-level example*: it pairs the visual observations at frame u of trajectory j with the robot's proprioceptive state at that frame, the task instruction associated with the trajectory, and the corresponding action chunk, written as

$$e_{j,u} = \big(v_{j,u}, s_{j,u}, \ell_j, A_{j,u}\big). \tag{2}$$

The visual observations $v_{j,u} = (I^{\text{top}}_{j,u}, I^{\text{wrist}}_{j,u})$ capture the scene from the top and wrist cameras. The state $s_{j,u}$ records the robot's proprioceptive information, ℓ_j specifies the demonstrated task, and $A_{j,u}$ contains the action chunk associated with that frame. Each shot thus pairs an observed situation and task instruction with a demonstrated response. A shot is distinct from a complete demonstration trajectory: multiple shots can be drawn from different frames of the same trajectory. Eq. 2 specifies the information available in each example; its encoding and use follow the evaluated model's input interface. As shown in Fig. 1, we retrieve demonstration examples based on visual similarity between the current observation and frames in the retrieval bank. Specifically, we encode the current top-camera image using a DINOv2 image encoder(Oquab et al., 2024). Candidate demonstration frames are encoded in the same way, and their embeddings are precomputed and stored in a task-specific retrieval bank. We then perform K-nearest-neighbor search to select the K candidate frames closest to the current observation in this embedding space. For each selected frame, we retrieve the complete example defined in Eq. 2, including both camera views, the proprioceptive state, the task instruction, and the associated action chunk. Thus, retrieval is based on top-camera image embeddings, while the retrieved shots contain the *full* multimodal information associated with the selected frames. The selected examples form the ordered demonstration context

$$E_t = \big(e_t^{(1)}, e_t^{(2)}, \ldots, e_t^{(K)}\big), \tag{3}$$

where $e_t^{(i)}$ denotes the i-th nearest example in the retrieval embedding space. We arrange the shots from left to right in decreasing order of retrieval similarity: $e_t^{(1)}$ is the most similar example, followed by progressively less similar examples. The current query is placed after the retrieved shots, as shown in Fig. 1. The demonstration context and the current query jointly form the complete context window:

$$C_t = \big(e_t^{(1)}, \ldots, e_t^{(K)}, q_t\big). \tag{4}$$

The demonstration examples provide observed situations paired with demonstrated actions, while the current query specifies the situation for which the policy must predict an action chunk. Thus, $W_{K,t}$ collects the inputs to the context-conditioned policy in Eq. 1; the action chunk $\hat{A}_t$ is its output. The shot count K measures the number of demonstration examples in this window. It does not count the current query, the number of distinct source trajectories, or the temporal length of an action chunk. We use $K = 0$ to denote prediction without accessible demonstration examples: the current query remains available, so the effective context window contains only q_t.

3.3 What Makes a Good In-Context Example?

Modality Composition. Each shot combines visual, linguistic, proprioceptive, and action information, but these components may contribute differently to action prediction. Does the policy require the full multimodal example, or can a subset of its components account for most of the benefit? Varying the available demonstration modalities while retaining the current query inputs allows us to examine which information supports context-conditioned behavior.

Observation–Action Correspondence. A complete shot associates a demonstrated action chunk with its original visual observations, robot state, and instruction. If the policy uses demonstrations to infer how to act in a given situation, preserving this correspondence may be important. Alternatively, useful information may reside in the demonstrated actions without depending on their exact pairing with observations. For $K \geq 2$, we examine this distinction by retaining the retrieved observations, states, and instructions while reassigning action chunks among shots:

$$\widetilde{E}_t = \Big(\big(v_t^{(i)}, s_t^{(i)}, \ell_t^{(i)}, A_t^{(\sigma(i))}\big)\Big)_{i=1}^{K}, \tag{5}$$

where σ is a permutation with no fixed points, such that $\sigma(i) \neq i$ for every shot. This intervention preserves the collection of action chunks while changing their association with the other components. It differs from reordering complete shots, which preserves these pairings, and from shuffling actions within a chunk, which changes its temporal structure.

Task Difficulty and Context Benefit. The usefulness of demonstration context may depend on the generalization demands of the query task. On tasks that the policy can already solve reliably, demonstrations may offer limited additional benefit. As tasks depart further from familiar configurations, relevant examples may provide information that helps the policy handle these changes. We therefore ask whether demonstration context yields larger performance improvements on tasks requiring greater generalization. To assess this relationship, we examine performance gains across predefined task difficulty levels, distinguishing the overall gains of the post-trained RICL system from the contribution of inference-time context measured with fixed model parameters.

Number of Shots. The number of examples determines how much demonstrated behavior is available for each prediction. Additional shots may provide useful coverage of relevant situations, but may also introduce redundant or less relevant information. It is therefore an empirical question whether increasing K consistently improves performance. We examine this question by varying the number of shots, separately from the temporal length of individual action chunks, and investigate whether the benefit of additional examples depends on task difficulty.

4 Experimental Setup

Model Selection. We select $\pi_{0.5}$-LIBERO and InternVLA-A1.5 as two representative backbones for studying in-context learning in VLAs. $\pi_{0.5}$-LIBERO provides a strong public checkpoint and an established LIBERO evaluation pipeline, making it a practical foundation for integrating and evaluating demonstration context. InternVLA-A1.5 unifies vision–language understanding, latent

foresight, and action generation, with foresight tokens supervised by a pretrained video model. Together, these backbones allow us to examine the use of context across distinct VLA architectures, including one with an explicit future-prediction component.

ICL Integration. Following the approach of RICL (Sridhar et al., 2025a), we extend $\pi_{0.5}$ and InternVLA-A1.5 to condition action predictions on retrieved demonstrations. Each RICL model is initialized from the same public LIBERO checkpoint used for its unmodified baseline, providing a pretrained visuomotor policy as the starting point for learning to use context. We post-train both models on the full LIBERO-40 demonstration set (2,000 trajectories across 40 tasks), using $K = 4$ examples per query. Examples are retrieved by DINOv2 similarity between top-view images, restricted to the query task, and excluding the query episode. We train $\pi_{0.5}$-RICL for 15,000 steps and InternVLA-A1.5-RICL for 30,000 steps. After the post-training, model parameters will remain fixed during ICL evaluation. More details are in Appendix B.

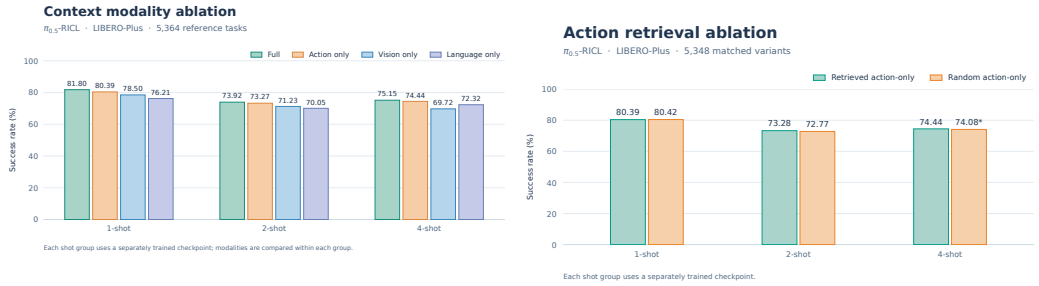
Datasets and Evaluation Tasks. All models are post-trained on the full LIBERO-40 demonstration set (Liu et al., 2023). Our evaluation includes the full LIBERO-Plus suite for selected experiments, a subset of LIBERO-Plus for broader comparative studies, a subset of LIBERO-Pro, and the full LIBERO-Mem suite. These evaluations examine context-conditioned performance on tasks and configurations *held out* from training. Demonstrations from these evaluation suites are used only as inference-time context, without parameter updates. The three evaluation suites share the LIBERO simulation ecosystem and compatible interfaces for robots, observations, and actions, enabling us to study diverse evaluation challenges within a controlled environment. LIBERO-Plus (Fei et al., 2026) introduces variants of LIBERO tasks across seven perturbation dimensions, providing a setting for examining whether demonstration context improves robustness to unseen configurations. LIBERO-Pro (Zhou et al., 2025) complements this evaluation with changes to objects, spatial arrangements, instructions, task goals, and environments; we evaluate a subset supported by the available demonstration bank. LIBERO-Mem (Chung et al., 2026) focuses on memory-dependent manipulation involving object motion, interaction sequences, object relations, and occlusion. It examines whether demonstration context helps with tasks where the current observation alone may be insufficient to determine the appropriate action. Retrieval banks for LIBERO-40, LIBERO-Plus, and LIBERO-Mem are constructed from their respective demonstration datasets. For LIBERO-Pro, matching demonstration trajectories are unavailable in the release used in our experiments. We therefore construct its retrieval bank from LIBERO-Plus demonstrations and restrict evaluation to tasks with corresponding or similar task settings. Table **??** summarizes the benchmark coverage, demonstration data, and evaluation scale, with the LIBERO-Plus counts referring to the full-suite evaluation. More details on subset selection and retrieval-bank construction are provided in Appendix C.

5 Experimental Results and Analysis

5.1 Which Modality Is Most Important in the Context?

Modality ablations. We compare full context with *action-only*, *vision-only*, and *language-only* variants for $\pi_{0.5}$-RICL on LIBERO-Plus, using separately trained 1-, 2-, and 4-shot checkpoints. Within each shot configuration, we keep the checkpoint, retrieved examples, and number of demonstration slots fixed, and mask the excluded demonstration modalities without additional training. The query inputs remain unchanged. The corresponding full-context result serves as the baseline for each shot configuration. Table 2 summarizes the modality settings. As shown in Figure 2a, action-only performs best among the single-modality variants in all three shot configurations. It achieves success rates of 80.39%, 73.27%, and 74.44% at 1, 2, and 4 shots, respectively, falling below the corresponding full-context baselines by only 1.42, 0.65, and 0.71 percentage points. The ordering of the other two modalities varies: vision-only outperforms language-only at 1 and 2 shots, whereas language-only performs better at 4 shots (72.32% versus 69.72%). These results indicate that actions are the most effective demonstration modality when retained alone in the evaluated configurations, consistently yielding performance closest to the corresponding full-context baseline. This comparison concerns single-modality retention and does not establish which modality would cause the largest performance drop if removed from full context. Since the shot configurations use separately trained checkpoints, differences across shot groups cannot be attributed solely to the number of context examples.

Does action retrieval matter? Given the strong performance of action-only context, we ask whether query-based retrieval provides an advantage over sampling actions from the same task. We compare



(a) Single-modality retention.

(b) Retrieved versus random action-only context.

Figure 2: **Modality contribution and action retrieval in $\pi_{0.5}$-RICL on LIBERO-Plus.** (a) Success rates with full context or only demonstration actions, vision, or language retained. (b) Success rates with retrieved or same-task randomly sampled actions; demonstration vision and language are masked in both conditions. Comparisons within each shot group use the same checkpoint, while the 1-, 2-, and 4-shot groups use separately trained checkpoints.

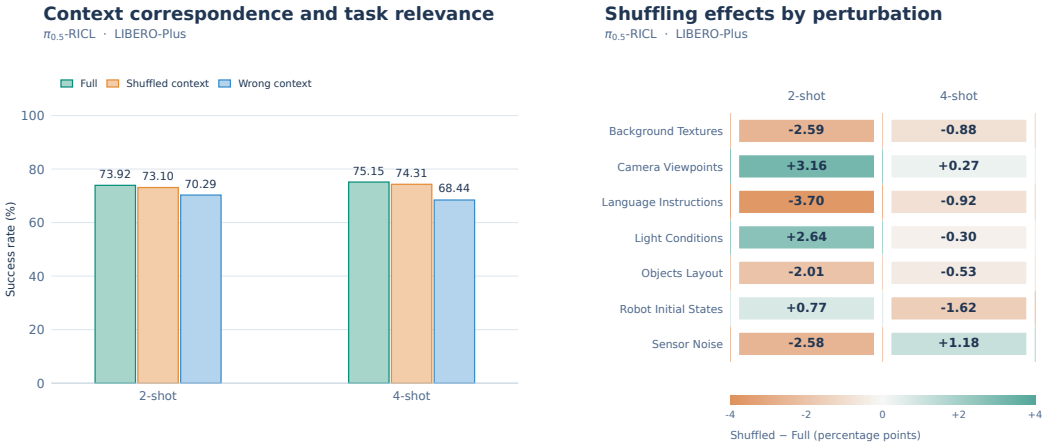


Figure 3: **Context correspondence and task relevance in $\pi_{0.5}$-RICL on LIBERO-Plus.** Left: success rates with full, shuffled, and wrong-task context for the separately trained 2- and 4-shot checkpoints. Right: the difference between shuffled and full context across seven perturbation categories, in percentage points; negative values indicate lower success after shuffling.

the retrieved action-only context with the randomly sampled action-only context for the 1-, 2-, and 4-shot $\pi_{0.5}$-RICL checkpoints on LIBERO-Plus. Retrieved actions come from the top-K demonstration frames selected for the query, whereas random actions come from demonstration timesteps sampled within the same task. Demonstration images and language are masked in both conditions, and the query inputs remain unchanged. As shown in Figure 2b, retrieved and random actions yield nearly identical success rates at 1 shot (80.39% versus 80.42%). Retrieval is higher by 0.51 percentage points at 2 shots (73.28% versus 72.77%). The plotted 4-shot values differ by 0.36 percentage points (74.44% versus 74.08%), but the random-action value lacks corresponding rollout records and is therefore provisional. Overall, the verified comparisons provide limited evidence of an additional benefit from query-based action retrieval in these settings. Actions sampled from the correct task appear to retain much of the performance of retrieved actions; this comparison does not establish that retrieval is unnecessary in other tasks or model configurations.

5.2 Does the Correct Mapping Between Actions and Observations Matter?

We compare full context with *shuffled context* on LIBERO-Plus. Starting from the same retrieved examples, we reassign complete action chunks across demonstration slots using a permutation with no fixed points. This preserves the retrieved observations and the collection of action chunks

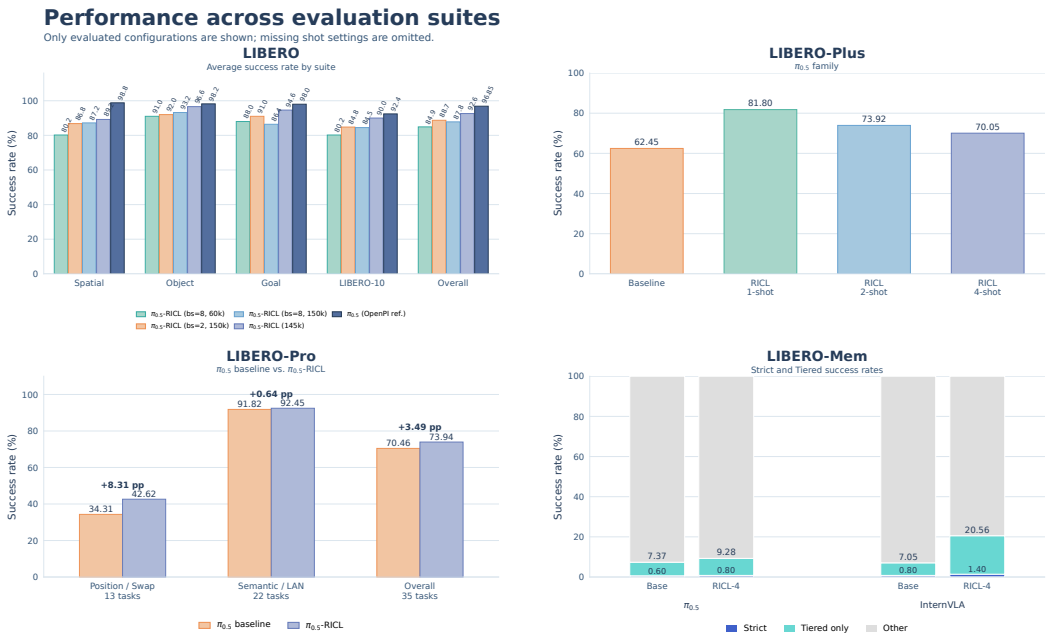


Figure 4: **Performance across four evaluation suites.** LIBERO shows the reported training settings and OpenPI reference; LIBERO-Plus shows the $\pi_{0.5}$ baseline and separately trained RICL shot configurations. LIBERO-Pro compares the $\pi_{0.5}$ baseline and RICL on the evaluated 13 Position/Swap and 22 Semantic/LAN tasks. LIBERO-Mem shows baseline and 4-shot RICL results for both model families; the dark segment denotes strict success, the colored segment above it denotes additional tiered successes, and the full colored height denotes tiered success.

while breaking their original observation–action pairings. We evaluate the 2- and 4-shot $\pi_{0.5}$-RICL checkpoints; a 1-shot context has no second slot with which to exchange actions. Shuffled context achieves 73.10% versus 73.92% with full context at 2 shots, and 74.31% versus 75.15% at 4 shots. The respective differences are 0.82 and 0.84 percentage points. Thus, breaking the observation–action pairings within the retrieved context is associated with a small change in the aggregate success rate for these two checkpoints. This does not imply that the policies ignore correspondence: the intervention retains the retrieved observations and the same collection of action chunks, and aggregate rates can conceal larger effects on particular tasks. We also replace the retrieved demonstrations with context drawn from an incorrect task's demonstration bank, while retaining the retrieval procedure. Wrong context yields 70.29% at 2 shots and 68.44% at 4 shots, compared with 73.92% and 75.15% for full context. The displayed gaps of 3.63 and 6.71 percentage points are larger than those for the shuffled context. In these evaluated settings, task relevance therefore appears more consequential than the precise slot-level observation–action pairing.

5.3 How Does ICL Perform on Tasks Requiring More Generalization?

Performance across evaluation suites. Figure 4 reports the configurations evaluated on each suite. On LIBERO, the strongest reported $\pi_{0.5}$-RICL setting reaches 92.6% overall, below the 96.85% OpenPI reference. On LIBERO-Plus, the separately trained 1-shot checkpoint achieves 81.80%, compared with 62.45% for the baseline; the 2-shot and 4-shot checkpoints achieve 73.92% and 70.05%, respectively. On the evaluated 35-task LIBERO-Pro subset, $\pi_{0.5}$-RICL improves overall success from 70.46% to 73.94%. The gain is concentrated in Position/Swap tasks (+8.31 percentage points), while Semantic/LAN changes by only +0.64 points. LIBERO-Mem remains challenging under the strict criterion: $\pi_{0.5}$ improves from 0.60% to 0.80%, and InternVLA from 0.80% to 1.40%. Under the tiered criterion, the corresponding changes are 7.37% to 9.28% and 7.05% to 20.56%. These results show that the effect of RICL depends on the evaluation suite and success criterion; comparisons across shot counts also reflect separately trained checkpoints rather than an inference-time shot ablation.

System gains across difficulty levels. We group LIBERO-Plus variants using the benchmark-provided difficulty labels (Fei et al., 2026). These labels are based on how many of four reference models—OpenVLA-OFT, π_0, π_0-fast, and UniVLA—successfully complete each variant: Level 1 is solved by all four models, Levels 2–4 by three, two, and one model, respectively, and Level 5 by none. The levels therefore reflect the reference models' outcomes rather than the performance of our $\pi_{0.5}$-RICL configurations. Figure 5 compares the available $\pi_{0.5}$-RICL configurations with the plain $\pi_{0.5}$ baseline. The 1-shot configuration exceeds the baseline at every level, with observed differences of 11.4, 13.9, 19.1, 18.0, and 20.8 percentage points from Levels 1 to 5. The corresponding differences are -1.0, 7.5, 11.3, 8.9, and 16.9 points for 2-shot, and 2.4, 5.1, 9.3, 6.6, and 10.0 points for 4-shot. The largest differences generally occur at higher difficulty levels, although the pattern is not monotonic: the difference decreases from Level 3 to Level 4 for all three shot configurations. Absolute success rates also decline toward Level 5, leaving more room for improvement on variants that the benchmark's reference models found challenging. Because the 1-/2-shot evaluations and the baseline/4-shot results do not use identical evaluation sets, these cross-configuration differences are descriptive rather than paired estimates of system gain. This analysis covers only the $\pi_{0.5}$ family; it does not establish whether the pattern extends to InternVLA.

5.4 Are More In-Context Demonstrations Always Better?

Separately trained shot configurations. For each model family, we compare RICL checkpoints trained with $K \in \{1, 2, 4\}$ and evaluated with the corresponding number of demonstrations. We include the original public baseline as a reference and examine zero-context inference separately in the fixed-checkpoint ablation below. Figure 5 shows the available $\pi_{0.5}$-RICL results across the five LIBERO-Plus difficulty levels. The 1-shot configuration has the highest success rate at every level, ranging from 91.9% at Level 1 to 59.9% at Level 5. Increasing from 2 to 4 shots does not yield a consistent improvement: 4-shot is higher at Level 1 (82.9% versus 79.5%), whereas 2-shot is higher at Levels 2–5. Thus, the separately trained configurations do not show a monotonic benefit from increasing K on this evaluation. Because each K uses a different checkpoint as well as a different number of inference demonstrations, these differences cannot be attributed to context length alone. The 4-shot results also come from a different evaluation set than the 1-/2-shot results, so the comparisons across configurations are descriptive.

Shot count and task difficulty. To examine the contribution of inference context within a single trained model, we group results from the fixed $\pi_{0.5}$-RICL 4-shot checkpoint by the LIBERO-Plus difficulty levels (Fei et al., 2026). We vary the accessible demonstrations at inference time over $K \in 0, 1, 2, 4$ while holding the checkpoint fixed. The appendix analysis in Figure 7 shows that $K = 2$ achieves the highest success rate at every level, with the largest improvement over $K = 0$ at Level 5. This fixed-checkpoint result indicates that providing context helps most on the hardest group in this evaluation, while increasing inference-time K to 4 does not provide a further benefit.

6 Conclusion

We systematically investigated what makes in-context demonstrations effective for VLAs. Through controlled interventions on demonstration content, observation–action correspondence, task generalization difficulty, and context length, we identified several important properties of current VLA in-context learning. We find that action tokens contribute substantially more to ICL performance than visual observations or language, while preserving the correct pairing between observations and actions is surprisingly not necessary for obtaining most of the observed ICL gains. We further show that ICL becomes more beneficial under more challenging generalization settings, yet providing more contextual information is not always helpful, as longer demonstrations can eventually reduce performance. Together, these results suggest that current VLA ICL does not simply operate by extracting a coherent state–action mapping from a small set of multimodal demonstrations. Instead, a substantial portion of the benefit may arise from action-side information and other contextual signals that do not require preserving the full multimodal structure of the demonstrations. These findings provide practical guidance for constructing more effective in-context examples, while also raising broader questions about what information VLAs actually infer from context and how future models can be trained to make more principled use of multimodal demonstrations.

REFERENCES

Skild AI. Introducing s1: In-context learning for robotics. August 2026. URL https://skild.ai/blogs/s1.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Jun Cen, Siteng Huang, Yuqian Yuan, Kehan Li, Hangjie Yuan, Chaohui Yu, Yuming Jiang, Jiayan Guo, Xin Li, Hao Luo, Fan Wang, Deli Zhao, and Hao Chen. Rynnvla-002: A unified vision-language-action and world model. *ArXiv*, abs/2511.17502, 2025. URL https://api.semanticscholar.org/CorpusID:283225904.

Nhat Chung, Taisei Hanyu, Toan Nguyen, Huy Le, Frederick Bumgarner, Duy Minh Ho Nguyen, Khoa Vo, Kashu Yamazaki, Chase Rainwater, Tung Kieu, Anh Nguyen, and Ngan Le. Rethinking progression of memory state in robotic manipulation: An object-centric perspective. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(5):3407–3415, mar 2026. doi: 10.1609/aaai.v40i5.37337. URL https://doi.org/10.1609/aaai.v40i5.37337.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: A progressive robustness benchmark for visual-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 38574–38583, 2026.

Letian Fu, Huang Huang, Gaurav Datta, Lawrence Yunliang Chen, Will Panitch, Fangchen Liu, Hui Li, and Kenneth Y. Goldberg. Icrt: In-context imitation learning via next-token prediction. *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5937–5944, 2024. URL https://api.semanticscholar.org/CorpusID:271974730.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A Vision-Language-Action model with Open-World generalization. *arXiv preprint arXiv:2504.16054*, 2025.

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

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, et al. Openvla: An open-source vision-language-action model. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713, 2025b.

Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, et al. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*, 2022.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Ho Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. In *Conference on Robot Learning*, 2024. URL https://api.semanticscholar.org/CorpusID:269635843.

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

Fangcheng Liu, Yeqing Shen, Anda Cheng, Weishi Mi, Chao Tang, Tingguang Li, Yong-Lu Li, and Yehui Tang. Embodied in-context learning for gpt-6 astra: Multimodal models as few-shot robot learners. Research preview, September 2026. URL https://mosi-ai.github.io/RoboICL-GPT6-Astra.github.io/. Code: https://github.com/Mosi-AI/RoboICL.

Haoxiang Ma, Junhao Cai, Xiaoxu Xu, Hao Li, Yuyin Yang, Yang Tian, Jiafei Cao, Hongrui Zhu, Zherui Qiu, Yuqiang Yang, et al. Internvla-a1. 5: Unifying understanding, latent foresight, and action for compositional generalization. *arXiv preprint arXiv:2607.04988*, 2026.

NVIDIA, Johan Bjorck, Nikita Cherniadev Fernando Castañeda, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, , et al. GR00T N1: An open foundation model for generalist humanoid robots. In *ArXiv Preprint*, March 2025.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=a68SUt6zFt. Featured Certification.

Anian Ruoss, Fabio Pardo, Harris Chan, Bonnie Li, Volodymyr Mnih, and Tim Genewein. LMAct: A benchmark for in-context imitation learning with long multimodal demonstrations. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=8aChcUzAhI.

Kaustubh Sridhar, Souradeep Dutta, Dinesh Jayaraman, and Insup Lee. RICL: Adding in-context adaptability to pre-trained vision-language-action models. In *9th Annual Conference on Robot Learning*, 2025a. URL https://openreview.net/forum?id=6AASPlloSt.

Kaustubh Sridhar, Souradeep Dutta, Dinesh Jayaraman, and Insup Lee. Regent: A retrieval-augmented generalist agent that can act in-context in new environments. In *International Conference on Learning Representations*, volume 2025, pp. 39385–39414, 2025b.

Generalist Team. Gen-1.5: Embodied foundation models are one-shot learners. *Generalist AI Blog*, 2026. https://generalistai.com/blog/gen-1.5.

Vitalis Vosylius and Edward Johns. Instant policy: In-context imitation learning via graph diffusion. In *International Conference on Learning Representations*, volume 2025, pp. 35011–35030, 2025.

Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Qian Zhu, He Sun, Yong Wang, Shuailei Ma, et al. A pragmatic vla foundation model. *arXiv preprint arXiv:2601.18692*, 2026a.

Wei Wu, Fangjing Wang, Fan Lu, He Sun, Shi Liu, Yunnan Wang, Yibin Yan, Yong Wang, Shuailei Ma, Xinyang Wang, et al. From foundation to application: Improving vla models in practice. *arXiv preprint arXiv:2607.06403*, 2026b.

Ryan Yu, Pushi Zhang, Starrick Liu, Brae Liu, Miracle Kang, Shalfun Li, Lights Shi, Ellie Ma, Ping Yang, Chris Pan, et al. Wall-oss-0.5 technical report. *arXiv preprint arXiv:2605.30877*, 2026.

Borong Zhang, Jiahao Li, Jiachen Shen, Yuhao Zhang, Yishuai Cai, Yuanpei Chen, Juntao Dai, Jiaming Ji, and Yaodong Yang. VLA-arena: An open-source framework for benchmarking vision-language-action models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=qjeCH95rbF.

Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, and Xipeng Qiu. Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11142–11152, 2024. URL https://api.semanticscholar.org/CorpusID:274992845.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *International Conference on Learning Representations*, volume 2026, pp. 60580–60606, 2026.

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

Table 1: Experimental settings for context quantity and quality. RICL-K denotes a checkpoint trained with K demonstrations. The public baseline and the masked zero-shot RICL condition use different model weights and are reported separately.

Setting	Checkpoint	Demonstration context at inference
Baseline	Public	No demonstrations.
1-shot	RICL-1	One retrieved example.
2-shot	RICL-2	Two retrieved examples.
4-shot	RICL-4	Four retrieved examples.
Masked 0/1/2/4-shot	Fixed RICL-4	$\pi_{0.5}$ only: expose 0, 1, 2, or 4 demonstration slots without changing model weights.
Wrong context	RICL-K	Replace retrieved examples with unrelated demonstrations.
Shuffled context	RICL-K	Reassign action chunks among the K retrieved examples, with no chunk retaining its original pairing ($K \geq 2$).
Random action	RICL-K	Replace demonstration actions with random actions, retaining the retrieved observations and instructions.

Table 2: Demonstration-modality ablations. "Yes" and "No" indicate whether the corresponding demonstration information is available to the policy. All RICL variants within a shot setting use the same checkpoint and retain the original query inputs. The principal comparison uses $K = 4$; the same interventions can be evaluated with the matching 1-shot and 2-shot checkpoints.

Setting	Vision	Language	Action	Checkpoint
Baseline	No	No	No	Public
Full context	Yes	Yes	Yes	RICL-K
Action only	No	No	Yes	RICL-K
Language only	No	Yes	No	RICL-K
Vision only	Yes	No	No	RICL-K

A Experimental Settings

Tables 1 and 2 summarize our context ablations. We vary context quantity using RICL checkpoints trained with 1, 2, or 4 demonstrations, and for $\pi_{0.5}$ additionally vary the number of visible demonstrations under a fixed RICL-4 checkpoint. We probe context quality by replacing retrieved demonstrations with unrelated examples, shuffling action chunks across demonstrations to break observation–action correspondence, or replacing demonstration actions with random actions. To isolate modality contributions, we further compare full-context conditioning against action-only, language-only, and vision-only variants while keeping the query inputs and checkpoint fixed within each shot setting. These interventions separate the effects of context amount, relevance, correspondence, and modality composition.

B More Details about ICL Integration

Shared context construction. Both RICL policies are initialized from their publicly released LIBERO checkpoints and are subsequently trained on the same LIBERO-40 demonstration set, consisting of 40 tasks from the Spatial, Object, Goal, and Long-Horizon suites, with 50 expert trajectories per task and 2,000 trajectories in total. No LIBERO-90, LIBERO-Plus, or LIBERO-Pro trajectories are added during RICL training.

We define one shot as a frame-level example

$$e_{j,u} = \left(I^{\text{top}}_{j,u}, I^{\text{wrist}}_{j,u}, s_{j,u}, \ell_j, A_{j,u}\right), \tag{6}$$

where $I^{\text{top}}_{j,u}$ and $I^{\text{wrist}}_{j,u}$ are the top- and wrist-camera observations, $s_{j,u}$ is the proprioceptive state, ℓ_j is the task instruction, and $A_{j,u}$ is the continuous action chunk starting at frame u. The action horizon is model-specific: $H = 10$ for $\pi_{0.5}$ and $H = 8$ for InternVLA-A1.5. Actions are represented as 7-DoF end-effector commands—three translation dimensions, three rotational dimensions, and

one gripper dimension—and are zero-padded to the 32-dimensional action interface used by both action experts.

Retrieval is performed using the top-camera image only. We resize the images to 224×224 and encode them with a frozen DINOv2 ViT-B/14 encoder. For a query frame $q = (j, u)$, the candidate set is restricted to frames from the same semantic task but from different demonstration episodes:

$$\mathcal{B}(q) = \{e_{j',u'} \mid \mathrm{task}(j') = \mathrm{task}(j),\ j' \neq j\}\,. \tag{7}$$

We compute the Euclidean distance between the query embedding and every candidate embedding,

$$d(q, e) = \left\|\phi(I_q^{\mathrm{top}}) - \phi(I_e^{\mathrm{top}})\right\|_2\,, \tag{8}$$

and select the K candidates with the smallest distances. The retrieved shots are ordered by increasing distance, such that $e^{(1)}$ is the nearest shot. Excluding the query episode prevents a model from receiving another frame of the same trajectory as an in-context example. During evaluation, the query rollout is not part of the offline retrieval bank, and retrieval is routed to the bank of the corresponding semantic base task. The main models use $K = 4$; the 1-shot and 2-shot results use separately trained models with $K = 1$ and $K = 2$, respectively.

$\pi_{0.5}$-RICL integration. We initialize $\pi_{0.5}$-RICL from the public $\pi_{0.5}$-LIBERO checkpoint. For every retrieved shot and for the current query, the visual observations are passed through the original SigLIP image encoder, while the task instruction is embedded by the original PaliGemma language pathway. Although the proprioceptive state is retained in the processed dataset, the adopted configuration uses `discrete_state_input=False`; consequently, state is not supplied to the $\pi_{0.5}$-RICL model.

The observation prefix is organized as

$$[P_1^{\mathrm{obs}}, \ldots, P_K^{\mathrm{obs}}, P_q^{\mathrm{obs}}]. \tag{9}$$

Each retrieved continuous action chunk is projected using the original action input projection of the action expert. We augment these action embeddings with two newly introduced components: a binary type embedding that distinguishes demonstration actions from query actions, and a rank embedding that identifies the retrieval position $i \in \{1, \ldots, K\}$. The resulting demonstration action blocks are

$$E_i^{\mathrm{act}} = W_{\mathrm{act}} A_i + E_{\mathrm{type}}(\mathrm{context}) + E_{\mathrm{rank}}(i). \tag{10}$$

Each demonstration action block attends to its associated observation block and to itself, whereas the noisy query-action suffix attends to the complete prefilled context, including all retrieved observations, retrieved actions, and the current query observation. The model predicts the flow field only for the query action chunk; retrieved actions are clean conditioning inputs and do not receive a training loss.

The reported $\pi_{0.5}$-RICL model additionally uses nearest-neighbor x_0 interpolation at inference:

$$\widehat{A} = (1 - \alpha)\widehat{A}_{\mathrm{model}} + \alpha A_{\mathrm{NN}}, \qquad \alpha = \exp(-\lambda \widetilde{d}_q), \tag{11}$$

where $\widetilde{d}_q$ is the normalized query-to-nearest-neighbor distance and $\lambda = 10$. This operation is applied after flow integration. In the main evaluation configuration, the retrieved gripper channel is masked to zero to prevent direct copying of a discrete gripper command.

InternVLA-A1.5-RICL integration. We initialize InternVLA-A1.5-RICL from the public InternVLA-A1.5-Libero checkpoint. Unlike $\pi_{0.5}$, InternVLA-A1.5 uses its native Qwen3.5 multimodal chat interface to encode each query or demonstration observation. Every prefix block contains the top- and wrist-camera images, the task instruction, and the robot state. The state is padded to 32 dimensions, scaled and discretized into 256 bins by the original InternVLA state encoder, and inserted into the textual Qwen input rather than projected as a continuous action-expert token.

A known demonstration action is represented through two complementary paths. First, it is tokenized by the original FAST action tokenizer and included as assistant tokens in the corresponding demonstration chat block. Second, the same continuous clean action chunk is passed through the original action input projection and time embedding of the Unified Expert with $t = 0$. Thus, FAST tokens provide a discrete language-side representation, while continuous action embeddings provide direct action-space evidence to the flow-matching expert.

Table 3: Implementation and optimization details of the two RICL policies. Both models use the same LIBERO-40 trajectories and the same task-restricted DINOv2 retrieval protocol.

Component	$\pi_{0.5}$**-RICL**	**InternVLA-A1.5-RICL**
Initialization	Public `pi05_libero` checkpoint	Public `InternVLA-A1.5-Libero` checkpoint
Training data	LIBERO-40: 40 tasks, 50 trajectories/task, 2,000 trajectories	
Retrieval	DINOv2 ViT-B/14 top-view features; exact L2 search; same task; different episode; ascending-distance top-K	
Context images	Top and wrist images; an additional right-image interface slot duplicates the top image	Top and wrist images through the native Qwen multimodal processor
Context language	Original PaliGemma language embeddings	Native Qwen3.5 chat tokens
Context state	Stored in the dataset but not used (`discrete_state_input=False`)	Padded, discretized into 256 bins, and inserted into the text input
Context action	Continuous action chunk projected into the action expert	Both FAST action tokens in Qwen and a continuous action chunk in the Unified Expert
New context parameters	Demonstration/query type projection and retrieval-rank projection	Context-role embedding and retrieval-rank embedding
Action horizon	10	8
Main number of shots	4; separately trained 1- and 2-shot checkpoints	4; separately trained 1- and 2-shot checkpoints
Training losses	Query flow-matching MSE	Query flow-matching loss, FAST auxiliary loss, and WAN foresight/video loss
Trainable modules	Language backbone, action expert, action input/output projections, time MLP, and new type/rank projections	Qwen visual and language components, Unified Expert, action projections, foresight tokens and video projections, and new role/rank parameters
Frozen modules	SigLIP image encoder	WAN DiT and VAE; the Qwen visual encoder remains trainable
Optimizer	AdamW, $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 10^{-10}, global gradient clipping 1.0	AdamW, $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 0.01, global gradient clipping 1.0
Learning-rate schedule	Cosine decay; 1,000-step warmup; peak 5×10^{-6}; final 5×10^{-7}	Cosine decay; 1,000-step warmup; peak 2.5×10^{-5}; final 2.5×10^{-6}
Training length	30,000-step schedule; main reported checkpoint at step 15,000	30,000 steps; main reported checkpoint at step 30,000
Batching	Batch size 2	Effective batch size 8; gradient checkpointing enabled
EMA	0.999	Not used
Inference	10 Euler denoising steps; x_0 nearest-action interpolation with $\lambda = 10$; execute 5 actions before replanning	10 flow-integration steps; no output-action interpolation in the final model
Main checkpoint	`pi05_libero_ricl_preserve_x0`, step 15,000	`internvla_isolated_v2_none_bs8`, step 30,000

The physical prefix layout is

$$[P_q, P_1, \ldots, P_K], \tag{12}$$

and the action-expert suffix is

$$[s, F_{1:50}, A_q, A_1, \ldots, A_K], \tag{13}$$

where $F_{1:50}$ denotes the 50 original foresight tokens. Context actions receive newly introduced context-role and retrieval-rank embeddings. The attention routing preserves the original query pathway while isolating demonstrations: A_i reads only P_i and its own action block, whereas A_q reads the query observation, all retrieved observation blocks, all retrieved continuous actions, and the foresight block. Prefix tokens never read action-expert suffix tokens. For the hybrid Qwen3.5 backbone, the convolutional and recurrent states of the Gated DeltaNet layers are reset between blocks, preventing demonstrations from communicating through the linear-attention state.

During training, query actions retain the original FAST auxiliary supervision, while demonstration FAST tokens are treated as known context. The model is jointly optimized with query flow-matching loss, FAST-token supervision, and the original WAN-based foresight/video loss. The final InternVLA result uses the native model output without x_0 interpolation.

Optimization. Table 3 summarizes the main differences. For the principal N-shot comparisons, $K = 1$, $K = 2$, and $K = 4$ correspond to separately trained checkpoints. The 0-shot condition is the unmodified public baseline rather than a context-masked RICL checkpoint.

C Datasets

C.1 Datasets and Retrieval-Bank Construction

Training demonstrations and evaluation tasks. We distinguish three statistical units throughout our experiments: *tasks* define manipulation problems, *demonstrations* are offline expert trajectories

used for training or retrieval, and *rollouts* are closed-loop evaluation episodes. These units are reported separately and are not combined into a single sample count.

The RICL policies are trained on demonstrations from four standard LIBERO suites: LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-10. Together, these suites contain 40 tasks. Each task provides 50 expert demonstration trajectories, producing

$$40 \text{ tasks} \times 50 \text{ trajectories/task} = 2{,}000 \text{ training trajectories}. \tag{14}$$

The same 2,000 trajectories are used for policy optimization and for constructing the offline retrieval banks.

LIBERO-Plus and LIBERO-Pro are used only for evaluation and contribute no additional training demonstrations. Unless stated otherwise, evaluation-time context is retrieved from the offline LIBERO-40 bank associated with the corresponding semantic base task rather than from the test rollout itself. LIBERO-Mem provides a separate collection of memory-oriented demonstrations and therefore uses an independent retrieval bank.

Trajectory preprocessing. The original LIBERO demonstrations are stored as task-specific HDF5 files. Each trajectory is converted into a sequence of synchronized observations, robot states, task instructions, and expert actions. At timestep u of demonstration j, we retain

$$r_{j,u} = \left(I_{j,u}^{\text{top}}, I_{j,u}^{\text{wrist}}, s_{j,u}, \ell_j, a_{j,u}\right), \tag{15}$$

where $I_{j,u}^{\text{top}}$ and $I_{j,u}^{\text{wrist}}$ denote the agent-view and eye-in-hand RGB images, respectively; $s_{j,u}$ is the proprioceptive state; ℓ_j is the task instruction; and $a_{j,u}$ is the expert action.

The proprioceptive state is stored as a seven-dimensional vector

$$s_{j,u} = \left[p_{j,u}^x, p_{j,u}^y, p_{j,u}^z, r_{j,u}^x, r_{j,u}^y, r_{j,u}^z, g_{j,u}\right], \tag{16}$$

containing the end-effector position, axis–angle orientation, and gripper state. The action is represented in the corresponding 7-DoF end-effector space:

$$a_{j,u} = \left[\Delta p_{j,u}^x, \Delta p_{j,u}^y, \Delta p_{j,u}^z, \Delta r_{j,u}^x, \Delta r_{j,u}^y, \Delta r_{j,u}^z, a_{j,u}^g\right]. \tag{17}$$

The first six dimensions describe translational and rotational control, while the final dimension controls the gripper.

The images are rotated by 180° to match the orientation used by the public LIBERO policy checkpoints. They are subsequently resized to 224×224 using aspect-ratio-preserving padding. The processed dataset contains two distinct camera observations: the top camera and the wrist camera. Some policy interfaces additionally require a right-camera field; in our processed data, this field is populated by copying the top-camera image and therefore does not constitute a third independent view.

For every trajectory, the processed arrays are stored in `processed_demo.npz`. Each file contains the following fields:

- the complete state sequence;
- the complete expert-action sequence;
- the resized top- and wrist-camera images;
- the duplicated right-camera interface field;
- the task instruction;
- precomputed DINOv2 image features; and
- metadata describing the image orientation, action convention, and source HDF5 file.

Frame-level expansion into ICL examples. The 2,000 trajectories are not treated as 2,000 indivisible training samples. Instead, each valid frame of a trajectory can serve as an independent query. For frame u of demonstration j, we define one frame-level shot as

$$e_{j,u} = \left(I_{j,u}^{\text{top}}, I_{j,u}^{\text{wrist}}, s_{j,u}, \ell_j, A_{j,u}\right), \tag{18}$$

where

$$A_{j,u} = (a_{j,u}, a_{j,u+1}, \ldots, a_{j,u+H-1}) \tag{19}$$

is an expert action chunk beginning at the selected frame.

The processed trajectories and retrieval indices are shared across model families, while the action horizon H is selected when a training sample is loaded. If the requested action chunk extends beyond the end of a trajectory, the translational and rotational action dimensions are padded with zeros, and the final valid gripper command is repeated.

This procedure converts every trajectory into multiple frame-indexed query–context samples without generating synthetic trajectories or adding new expert behavior. The model therefore observes only actions and observations from the original demonstrations, while the in-context examples are selected dynamically according to the current query frame.

DINOv2 retrieval representation. We use a frozen DINOv2 ViT-B/14 image encoder to construct the retrieval representation. Retrieval is based only on the top-camera image. The wrist image, robot state, task instruction, and expert action are not used to calculate the retrieval distance.

For every processed frame, the top-camera image is encoded as

$$z_{j,u} = \phi_{\mathrm{DINOv2}}\left(I_{j,u}^{\mathrm{top}}\right), \tag{20}$$

where ϕ_{DINOv2} denotes the frozen DINOv2 encoder. We retain the selected normalized patch features and concatenate them into a fixed-dimensional representation. The embeddings are computed once during preprocessing and stored alongside the original observations.

Although retrieval is performed using only $z_{j,u}$, selecting a frame returns the complete shot defined in Equation 18. Consequently, a retrieved example contains both camera observations, the state, the instruction, and the associated action chunk.

Task-specific retrieval banks. We construct an independent retrieval bank for each semantic task m:

$$\mathcal{B}_m = \{(z_{j,u}, e_{j,u}) \mid \mathrm{task}(j) = m\}. \tag{21}$$

The task identity is obtained from the source LIBERO suite and HDF5 file. Keeping separate banks prevents a visually similar but semantically unrelated task from being selected as context.

For example, a query from a "place the bowl on the plate" task retrieves only from demonstrations of the same semantic task. It cannot retrieve a frame from a different task such as "open the drawer", even if the two frames have similar backgrounds or robot configurations.

Same-task, cross-episode retrieval. Let $q = (j, u)$ denote a query at frame u of demonstration episode j. Candidates are restricted to the same semantic task but must come from different demonstration episodes:

$$\mathcal{C}(q) = \left\{(z_{j',u'}, e_{j',u'}) \in \mathcal{B}_{\mathrm{task}(j)} \;\middle|\; j' \neq j\right\}. \tag{22}$$

The condition $j' \neq j$ excludes the complete query episode, not only the current query frame. This prevents temporally adjacent observations from the same trajectory from becoming in-context demonstrations.

For every candidate, we calculate the Euclidean distance

$$d\left(q, e_{j',u'}\right) = \left\|z_{j,u} - z_{j',u'}\right\|_2. \tag{23}$$

The candidates are sorted by increasing distance, and the K nearest frames are retained:

$$C_{K,q} = \underset{\substack{\mathcal{S}\subseteq\mathcal{C}(q)\\|\mathcal{S}|=K}}{\arg\min} \sum_{e\in\mathcal{S}} d(q, e). \tag{24}$$

We write the ordered result as

$$C_{K,q} = \left(e_q^{(1)}, e_q^{(2)}, \ldots, e_q^{(K)}\right), \tag{25}$$

where

$$d\left(q, e_q^{(1)}\right) \leq d\left(q, e_q^{(2)}\right) \leq \cdots \leq d\left(q, e_q^{(K)}\right). \tag{26}$$

Thus, the first context slot always contains the nearest retrieved shot.

Precomputed retrieval metadata. Retrieval is precomputed for every training frame. The result associated with each demonstration is stored in `indices_and_distances.npz`, which contains:

- the episode and frame index of every query;
- the episode and frame indices of its K retrieved shots; and
- the corresponding retrieval-distance metadata.

This avoids running DINOv2 and nearest-neighbor search repeatedly during training. At data-loading time, the stored indices are used to recover the complete observations and action chunks from the appropriate `processed_demo.npz` files.

For training-time configurations that use distance-dependent action weighting, the distances are normalized using statistics computed from the training retrieval bank. Similarity weights are then constructed as

$$w_i = \exp(-\lambda \widetilde{d}_i), \tag{27}$$

where $\widetilde{d}_i$ is the normalized distance and λ controls the strength of the distance weighting. The retrieved identities and their order are determined by the original DINOv2 distance, independently of this optional weighting.

Online retrieval during evaluation. During closed-loop evaluation, the current top-camera observation is processed with the same resizing, normalization, and DINOv2 encoder used during preprocessing. The task instruction is mapped to the corresponding semantic base task, and retrieval is performed only within that task's offline bank.

The evaluation rollout is not inserted into or used to update the retrieval bank. Consequently, test observations never retrieve future states or actions from the current rollout. The retrieval bank, its embeddings, and all model parameters remain fixed throughout evaluation.

For perturbation benchmarks derived from the standard LIBERO tasks, such as LIBERO-Plus, the perturbed test environment is matched to its underlying semantic base task. The retrieved demonstrations therefore come from the corresponding offline base-task bank, while the query observation is obtained from the perturbed environment.

Shot count. The number of shots K denotes the number of retrieved frame-level examples, not the number of complete trajectories and not the temporal length of the action chunk. The current query is always present and is not counted as a shot. The resulting context window is

$$W_{K,q} = \left(e_q^{(1)}, e_q^{(2)}, \ldots, e_q^{(K)}, q\right). \tag{28}$$

The main experiments use $K = 4$. The 1-shot and 2-shot experiments use $K = 1$ and $K = 2$, respectively. The 0-shot condition contains only the current query and does not access the retrieval bank.

Dataset coverage. Table 4 summarizes the datasets used for training, retrieval, and evaluation. Demonstration counts and rollout counts are reported separately because they correspond to different experimental units.

LIBERO-40 evaluation. LIBERO-40 contains the same 40 semantic tasks used to construct the training and retrieval data. The evaluation covers all four suites: Spatial, Object, Goal, and Long-Horizon. We conduct 50 closed-loop rollouts for each task, resulting in

$$40 \text{ tasks} \times 50 \text{ rollouts/task} = 2{,}000 \text{ rollouts} \tag{29}$$

per model and experimental condition. Although the semantic tasks are shared with the training set, evaluation uses simulator-generated initial states and closed-loop execution rather than replaying the expert trajectories.

LIBERO-Plus evaluation. LIBERO-Plus contains 10,030 perturbation variants derived from the same 40 semantic base tasks used by LIBERO-40. The benchmark covers four base suites and seven perturbation categories:

Table 4: Training and evaluation data used in our experiments. Demonstrations are offline expert trajectories, whereas rollouts are closed-loop evaluation episodes. Evaluation volumes are reported per model and experimental condition unless explicitly marked as totals.

Benchmark	Role	Scope used	Data/evaluation volume	Capability evaluated
LIBERO-40	Policy training, retrieval-bank construction, and in-distribution evaluation	40 tasks from Spatial, Object, Goal, and Long-Horizon suites	2,000 expert trajectories; 2,000 evaluation rollouts per condition	Language-conditioned manipulation across spatial relations, object identities, goal semantics, and long-horizon task compositions
LIBERO-Plus	Zero-shot robustness evaluation	10,030 perturbation variants derived from the 40 LIBERO base tasks	No additional demonstrations; one rollout per variant in a full condition	Robustness to changes in camera viewpoint, robot state, language, illumination, background, sensor noise, and object layout
LIBERO-Pro	Structured out-of-distribution evaluation	13 Position/Swap and 22 Semantic/LAN task–perturbation pairs	No additional demonstrations; 35 pairs $\times$ 50 rollouts = 1,750 rollouts per condition	Controlled generalization under object-position and semantic/language changes
LIBERO-Mem	Memory-oriented retrieval and evaluation	10 non-Markovian tasks	1,200 offline trajectories: 100 training and 20 validation trajectories per task	Ordered subgoals and object-centric memory involving motion, sequence, relation, and occlusion

Table 5: Fixed LIBERO-Plus subsets used for paired ablations.

Subset	Variants	Fraction	Primary use
C20 legacy	5,364	53.48%	Early $\pi_{0.5}$ context-length and modality ablations
S10-v1	2,757	27.49%	Fixed paired comparisons between policy families
N5	1,396	13.92%	Paired InternVLA 1-/2-/4-shot and modality ablations

1. camera viewpoints;
2. robot initial states;
3. language instructions;
4. light conditions;
5. background textures;
6. sensor noise; and
7. object layouts.

These variants define new evaluation environments but do not provide additional expert action demonstrations. In the full evaluation, every variant is executed once per model and condition, producing 10,030 closed-loop episodes per complete condition.

For every perturbation variant, the benchmark task is mapped to its semantic LIBERO-40 base task. The current perturbed observation is used as the DINOv2 query, while the retrieved examples come from the offline demonstration bank of the corresponding unperturbed semantic task.

Fixed LIBERO-Plus subsets. Evaluating all context-length and modality ablations on all 10,030 variants is computationally expensive. We therefore use fixed reduced subsets for paired comparisons. These subsets contain evaluation environments only and are never used for training.

All compared models and context conditions use exactly the same variant identities within a subset. Environment seeds, reset policies, timeout handling, and failure handling are also held fixed, enabling paired comparisons instead of comparisons between independently sampled tasks.

The S10-v1 subset contains at most ten variants from each $(\text{base task}, \text{perturbation category})$ group. Selection uses a deterministic SHA-256 ordering and approximately preserves the official difficulty distribution.

The N5 subset contains at most five variants from each of the 280 $(\text{base task}, \text{perturbation category})$ groups. Selection is deterministic and outcome-blind. It first prioritizes coverage of official difficulty levels 1–5 and then fills remaining positions using a fixed SHA-256 ranking with seed `20260923`.

The final subset contains 1,396 variants: 349 from each of the four suites and approximately 200 from each perturbation category.

LIBERO-Pro evaluation. The complete LIBERO-Pro benchmark contains task perturbations along multiple dimensions. Our reported experiments focus on two controlled forms of distribution shift:

- **Position/Swap**, which changes the positions or arrangement of task-relevant objects; and
- **Semantic/LAN**, which changes or paraphrases the language instruction while retaining the intended task.

The selected evaluation scope contains 13 Position/Swap pairs and 22 Semantic/LAN pairs, giving 35 task–perturbation pairs in total. Each pair is evaluated using 50 closed-loop rollouts:

$$13 \times 50 + 22 \times 50 = 650 + 1{,}100 = 1{,}750 \tag{30}$$

evaluation episodes per model and condition. LIBERO-Pro provides no additional training demonstrations in our setup, and no model parameter is updated on LIBERO-Pro tasks.

LIBERO-Mem data and evaluation. LIBERO-Mem contains 10 long-horizon, non-Markovian tasks designed to evaluate object-centric memory and ordered subgoal execution. Its tasks cover four memory-related dimensions: motion, sequence, relation, and occlusion.

The benchmark provides 120 offline trajectories for each task, including 100 training trajectories and 20 validation trajectories. The complete offline collection therefore contains

$$10 \text{ tasks} \times 120 \text{ trajectories/task} = 1{,}200 \text{ trajectories.} \tag{31}$$

LIBERO-Mem trajectories are processed using the same frame extraction, image preprocessing, DINOv2 encoding, and task-specific retrieval procedure described above. However, they form a separate retrieval bank and are never mixed with the standard LIBERO-40 demonstrations.

For formal evaluation, every model condition is evaluated with 50 rollouts for each of the 10 tasks, giving

$$10 \text{ tasks} \times 50 \text{ rollouts/task} = 500 \text{ rollouts} \tag{32}$$

per condition. Across four reported model conditions, this corresponds to 2,000 evaluation rollouts in total. Interface-validation episodes, smoke tests, retry duplicates, and incomplete runs are excluded from the formal aggregate.

D More Detailed Analysis

Is the effect consistent across perturbations? The perturbation breakdown reveals variation hidden by the small aggregate shuffled-context differences. At 2 shots, shuffling is associated with decreases of 3.70 percentage points for Language Instructions and 2.59 points for Background Textures, but increases of 3.16 points for Camera Viewpoints and 2.64 points for Light Conditions. At 4 shots, the category differences are smaller in magnitude: the largest decrease is 1.62 points for Robot Initial States, while Sensor Noise shows a 1.18-point increase. The direction of the shuffling effect therefore varies across perturbation categories, and its small aggregate magnitude should not be read as a uniform lack of sensitivity to observation–action correspondence.

Inference-time shot ablation. We evaluate a fixed 4-shot $\pi_{0.5}$-RICL checkpoint on LIBERO-Plus with $K \in \{0, 1, 2, 4\}$ demonstrations accessible at inference time. Unused demonstration slots are masked, and $K = 0$ leaves only the query inputs accessible. Across the 10,030 evaluated variants, success rates are 69.19%, 72.11%, 73.27%, and 70.05%, respectively; thus, $K = 2$ performs best among the tested settings, while increasing to $K = 4$ reduces success by 3.22 percentage points. Figure 6 shows that this effect varies across perturbation categories. Relative to $K = 0$, $K = 2$ improves success most on language-instruction variants (+12.0 percentage points), whereas it reduces success on sensor-noise variants (-1.5 points). The benefit of additional inference demonstrations therefore depends on the perturbation, and more accessible shots do not consistently improve this fixed checkpoint.

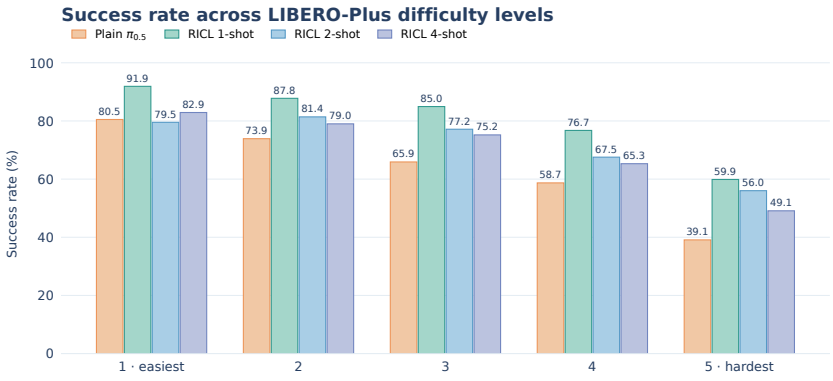


Figure 5: $\pi_{0.5}$ **performance across LIBERO-Plus difficulty levels.** Success rates of the plain $\pi_{0.5}$ baseline and separately trained 1-, 2-, and 4-shot $\pi_{0.5}$-RICL configurations. LIBERO-Plus defines difficulty by the number of four reference models that succeed on each variant, from four at Level 1 to none at Level 5 (Fei et al., 2026). The 1-/2-shot and baseline/4-shot results do not use identical evaluation sets; differences between their bars should not be interpreted as paired improvements.

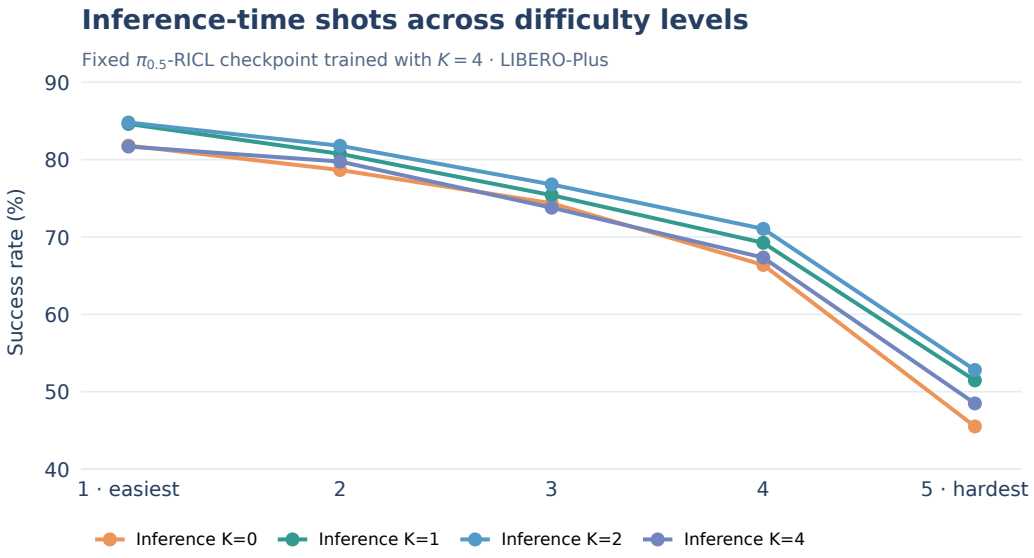


Figure 6: **Inference-time shot ablation across LIBERO-Plus perturbation categories.** Success rates under seven perturbation categories for a fixed $\pi_{0.5}$-RICL checkpoint trained with four demonstration slots. Only the number of accessible demonstrations at inference time changes ($K = 0, 1, 2, 4$). All four conditions evaluate the same 10,030 variants.

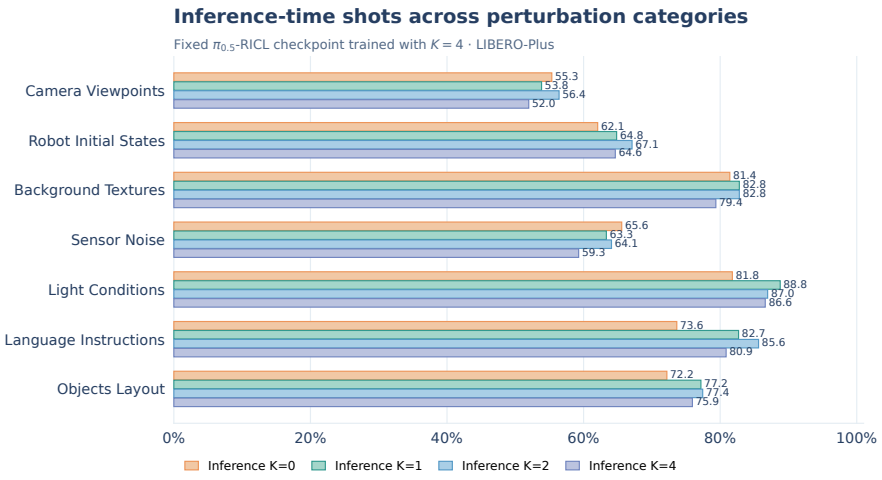


Figure 7: **Inference-time shot ablation across LIBERO-Plus difficulty levels.** Success rates for the same fixed 4-shot $\pi_{0.5}$-RICL checkpoint with $K = 0, 1, 2, 4$ accessible demonstrations at inference time. The difficulty levels follow the benchmark-provided labels (Fei et al., 2026). The plot includes 9,909 variants with assigned difficulty levels; 121 variants without labels are excluded.