PROACTIVEVLA: AUGMENTING EMBODIED MEMORY THROUGH PROACTIVE ENVIRONMENT EXPLORATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Rapid adaptation to a new environment requires a robot to acquire useful knowledge about local objects, states, and interactions from limited experience. Systems that combine a reasoning agent with a frozen vision-language-action model (VLA) can adapt through execution feedback and memory, making the choice of experience central to their effectiveness. Repeated practice of a target task may refine a familiar solution while leaving other interactions relevant to changed conditions untested. We introduce ProactiveVLA, which uses proactive environment exploration to acquire reusable knowledge for deployment-time adaptation. After completing an initial task, the agent allocates the remaining interaction budget to self-proposed goals covering object affordances, state-changing interactions, and compositions of interactions. It verifies execution outcomes and consolidates both task-directed and exploratory experience into memory that guides subsequent planning and control. ProactiveVLA outperforms the baselines under the same turn budget on LIBERO-Pro and RoboCasa365 Composite-Seen. On LIBERO-Pro Goal-T, with at most one VLA primitive invocation allowed during evaluation, ProactiveVLA completes 48% of instances, compared with 19% for the Harness VLA task-refinement baseline.

1 INTRODUCTION

A robot deployed in an unfamiliar environment must use its existing capabilities with new objects, layouts, and task requirements. Vision-language-action models (VLAs), including OpenVLA and $\pi_{0.5}$, provide pretrained manipulation capabilities learned from broad robot data (Kim et al., 2025; Black et al., 2025a). However, possessing these capabilities does not establish how they work in a particular deployment environment. The robot needs experience with local object uses, action outcomes, and interaction sequences to guide its decisions. Grounding decisions in available skills and environment feedback helps connect language-level knowledge to feasible robot behavior (Ichter et al., 2023; Huang et al., 2023). With limited opportunities for interaction, rapid adaptation depends on acquiring information that supports task execution beyond the specific conditions already encountered.

Recent advances in agentic robotics use language models to organize perception, planning, and control through executable robot interfaces (Liang et al., 2023; Singh et al., 2023; Fu et al., 2026; Zhang et al., 2026b). Such systems make experience acquisition an explicit design choice, since the agent determines which interactions to attempt and what knowledge to preserve. Harness VLA provides a state-of-the-art agentic framework for adapting frozen VLAs, exposing the VLA as a callable manipulation primitive and combining it with analytic control and execution memory. During its exploratory bootstrapping phase, Harness VLA searches for a working solution on a reference task instance and refines its execution memory across attempts (Zhang et al., 2026b). We refer to experience acquisition that keeps the provided task objective fixed while improving its execution as *task refinement*. Although this can improve a working solution, the resulting experience may remain concentrated around the same task requirements. For example, practicing placement with an unobstructed target can refine grasping and release but may provide little direct evidence about clearing space when the target becomes obstructed. Handling this change may require an interaction absent from the successful trajectory. Improving a familiar solution and expanding the conditions under which the robot can act are therefore different uses of the same interaction budget.

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.

Self-directed play already provides a route to acquiring reusable robot skills, as demonstrated by RATS (Zhang et al., 2026a). For rapid adaptation of a VLA-based agentic system, we focus on using limited deployment-time interaction to acquire knowledge relevant to the local environment and its tasks. This knowledge concerns the roles objects can serve, the changes an interaction can produce, and the ways interactions can be combined. Under a limited interaction budget, actively acquiring diverse and reusable environment knowledge can improve adaptation to changes in tasks and execution conditions beyond practicing a single solution.

We introduce ProactiveVLA, an adaptation framework that enables agentic VLA systems to proactively acquire and organize environment knowledge through interaction. The agent first tries the target task to obtain concrete experience with the local environment and controller. It then uses the remaining budget to propose and execute goals at three levels, covering object affordances, state-changing interactions, and interaction compositions. Visual outcome checks connect each proposal to observed execution evidence. A memory-management process organizes experience within scenes, synthesizes candidate knowledge across scenes, and uses an independent critic to review proposed rules before they guide subsequent execution. Both initial task experience and exploration outcomes contribute to this memory. Adaptation takes place through the agent's acquired knowledge while the VLA remains frozen. The experiments show that ProactiveVLA outperforms Harness VLA under the same adaptation-turn budget on LIBERO-Pro. Under the same limited VLA primitive invocation budget, ProactiveVLA achieves higher task success, with an advantage when only a small number of VLA calls are available.

Our contributions are threefold. First, we study deployment-time adaptation as the acquisition of reusable environment knowledge under a fixed interaction budget. Second, we develop a task-guided proactive exploration method that couples three levels of goal generation with outcome verification and memory consolidation. Third, we evaluate ProactiveVLA on LIBERO-Pro and RoboCasa365 Composite-Seen, covering controlled task and spatial variations as well as multi-step household tasks, and analyze how exploration structure and memory consolidation affect adaptation.

2 Related Work

Vision-language-action models and generalization. Vision-language-action models learn language-conditioned visuomotor policies from large collections of robot trajectories and vision-language data. RT-2 introduced the VLA formulation by jointly representing language and robot actions, while Open X-Embodiment expanded cross-embodiment robot data at scale (Zitkovich et al., 2023; O'Neill et al., 2024). OpenVLA provided an open generalist VLA, and subsequent models such as π_0 and $\pi_{0.5}$ scaled continuous action generation and heterogeneous robot training (Kim et al., 2025; Black et al., 2025b;a). These models obtain strong manipulation capabilities from pretraining, but robustness can decrease when deployment changes task bindings, object arrangements, or compositional requirements. LIBERO-Pro explicitly evaluates such changes through controlled perturbations of LIBERO tasks (Liu et al., 2023a; Zhou et al., 2026), while RoboCasa365 provides diverse multi-step household manipulation tasks in kitchen environments (Nasiriany et al., 2026).

Agentic adaptation around robot policies. Language-model agents provide another route to generalization by composing perception and control interfaces around pretrained robot capabilities. Language-model agents interleave reasoning with actions and environment feedback (Yao et al., 2023). In robotics, this interaction connects high-level decisions to perception and control interfaces. Code as Policies and ProgPrompt use executable programs or program-like plans to connect language-model reasoning with robot APIs (Liang et al., 2023; Singh et al., 2023). More recent systems increasingly treat the policy as one component inside a persistent execution harness. Harness VLA exposes a frozen VLA as a contact-rich manipulation primitive and combines it with high-level agent reasoning, analytic control, and execution memory (Zhang et al., 2026b). Recent agentic systems further explore self-improvement through interaction feedback, skill discovery, runtime recovery, and experience accumulation (Lu et al., 2026; Xiao et al., 2026; Zhang et al., 2026a; Ding et al., 2026). ProactiveVLA studies adaptation efficiency under a limited interaction budget, focusing on what experience should be acquired from a new environment. It allocates this budget to acquiring reusable environment knowledge rather than concentrating additional experience on a single task.

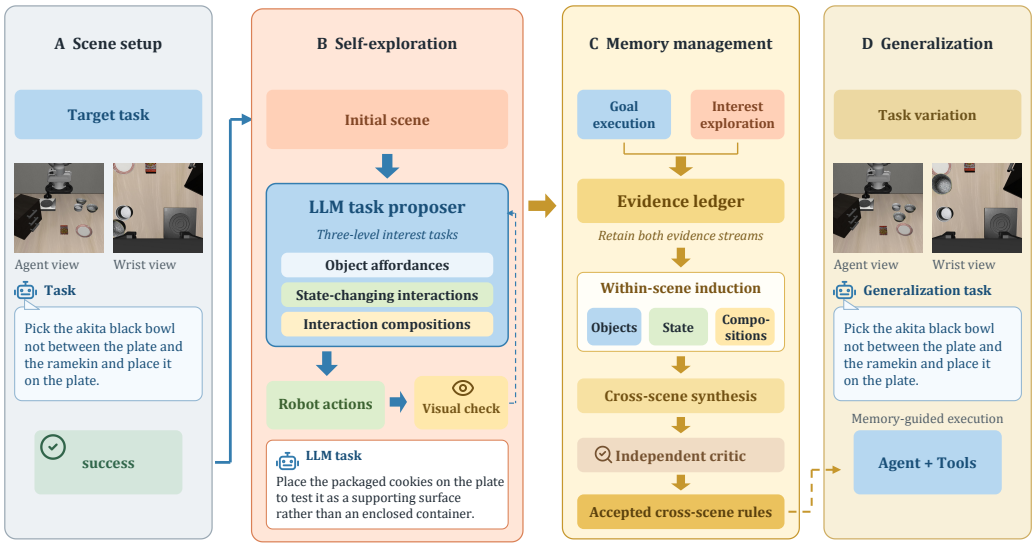


Figure 1. Overview of ProactiveVLA. After completing the environment-provided task, the agent uses the remaining interaction budget to explore scene-related goals involving object affordances, state-changing interactions, and interaction compositions. An independent visual checker assesses exploratory outcomes and supplies feedback for continued exploration. Memory management preserves the complete interaction records, organizes experience within scenes, synthesizes knowledge across scenes, and independently reviews candidate rules. Accepted rules form a frozen global memory that guides execution across tasks and variants.

Autonomous exploration and reusable experience. Prior work has also studied how agents can autonomously acquire experience beyond a single externally specified task. Plan2Explore uses task-agnostic exploration to learn a world model for later adaptation (Sekar et al., 2020). ELLM uses language models to propose semantically meaningful exploration goals, showing that language-guided exploration can improve the usefulness of pretraining experience (Du et al., 2023b). Voyager combines automatic curriculum generation, environment interaction, and a persistent skill library for open-ended embodied learning (Wang et al., 2024). More closely related to robot manipulation, recent agentic systems use self-directed interaction to discover reusable skills or behavioral knowledge that can later support downstream tasks (Lu et al., 2026; Zhang et al., 2026a). Experience can also be consolidated through reflection and verification. Reflexion retains verbal reflections in episodic memory, while ReasoningBank distills reasoning strategies from self-judged successful and failed attempts and updates memory across tasks (Shinn et al., 2023; Ouyang et al., 2026). REFLECT summarizes multimodal robot experience to diagnose failures and guide subsequent correction (Liu et al., 2023b). ProactiveVLA shares the premise that interaction should create reusable knowledge, but studies a budgeted preparation problem.

3 METHODOLOGY

ProactiveVLA supports rapid adaptation by acquiring environment experience, consolidating it into a global memory, and using that memory to guide subsequent behavior. A high-level agent interacts with the environment through a set of robot primitives, such as moving to a target pose, adjusting the robot posture, controlling the gripper, and VLAs. Each adaptation instance consists of a scene and an environment-provided task, with a shared budget of B interaction turns for task execution and autonomous exploration. The method comprises three stages, illustrated in Figure 1. Exploration collects task-directed and self-directed experience, memory management consolidates the evidence across adaptation scenes, and deployment uses the resulting global memory.

3.1 PROACTIVE ENVIRONMENT EXPLORATION

The agent first attempts the target task g_0. Following some attempts at b_{init} turns, it retains the task-execution experience and allocates the remaining $B - b_{\text{init}}$ turns to exploring other scene-related goals. A language-model proposer uses the observed scene and accumulated execution feedback to generate these goals at three levels.

Object affordances. The first level explores atomic or local interactions that reveal how individual objects can be used. Examples include grasping a bowl, opening a cabinet door, or pressing a faucet control. These interactions provide concrete evidence about the roles and operations supported by objects in the current scene. The goal at this level is to establish basic object–action relations that later exploration can build upon.

State-changing interactions. The second level explores larger but reasonable changes to the scene to understand its functional semantics. The agent autonomously proposes goals involving storage, placement, opening, closing, or other meaningful state changes. These interactions provide evidence about how objects and fixtures can be used to establish different scene states. For example, a Level-2 task may ask the agent to "place the bowl into the already opened top drawer and verify that the bowl is stably contained inside."

Interaction compositions. The third level explores how multiple verified interactions can be composed to accomplish a more complex goal. For example, the agent may first pick up a vegetable, rinse it under running water, transfer it into a bowl, and then complete the final placement. Such goals require the agent to reason about execution order and intermediate conditions connecting the component skills.

Visual verification and continued exploration. Vision-language models have been studied as success detectors for language-specified tasks (Du et al., 2023a). An independent visual checker assesses each attempted exploration goal using its description and visual observations of the execution outcome. Its assessment provides feedback for subsequent goal selection and strategy adjustment. The agent retains the complete interaction records, including successful, failed, and uncertain attempts, together with recovery behavior and visual assessments. An ordinary failure leads to a change of strategy and further exploration from the current state without resetting the environment. Both continued attempts and recovery interactions contribute experience within the remaining budget.

3.2 MEMORY MANAGEMENT

After exploration, ProactiveVLA consolidates experience from both the provided task and self-proposed goals. The process consists of lossless archival, experience atomization, within-scene induction, cross-scene synthesis, independent evidence review. Its output is a global memory $\mathcal{M}_{\text{global}}$ containing accepted knowledge synthesized across the adaptation scenes.

Lossless archival and experience atomization. The complete recorded trajectories are first preserved in an evidence archive $\mathcal{E}$. This archive retains successes, failures, uncertain outcomes, and recoveries from both experience sources, together with their recorded actions and visual assessments. Experience atomization then organizes these trajectories into individual experience units for subsequent induction. Each unit remains associated with its source interaction evidence, allowing later summaries and candidate rules to be reviewed against the original records.

Within-scene induction and cross-scene synthesis. Within each scene, the experience units are organized into three forms of knowledge corresponding to the exploration levels. Object-level knowledge summarizes observed object uses and relations. Action-level knowledge summarizes state-changing operations and the conditions observed during their execution. Compositional knowledge summarizes interaction sequences and their intermediate conditions. Cross-scene synthesis then combines the scene-level summaries into candidate rules about object use, action effects, and interaction composition. These candidates consolidate experience into knowledge intended to guide different tasks and variants.

Algorithm 1 Proactive exploration and global memory construction with ProactiveVLA

Require: instances $\{(e_i, g_i)\}_{i=1}^{N}$, interaction budget B per instance, agent $\mathcal{A}$, robot primitives $\mathcal{P}$

Initialize interaction records $\mathcal{T} \leftarrow \varnothing$
for each instance (e_i, g_i) **do**
    Attempt g_i using $\mathcal{A}$ and $\mathcal{P}$
    Retain the complete task-execution record in $\mathcal{T}$
    while interaction budget remains **do**
        Propose an exploration goal using the three-level objective and accumulated feedback
        Execute the goal using $\mathcal{A}$ and $\mathcal{P}$
        Assess the outcome with the visual checker
        Append the interaction record and assessment to $\mathcal{T}$
        Update the exploration strategy from the accumulated feedback
    end while
end for
$\mathcal{E} \leftarrow \mathrm{LosslessArchive}(\mathcal{T})$
$\mathcal{U} \leftarrow \mathrm{AtomizeExperience}(\mathcal{E})$
for each adaptation scene s represented in $\mathcal{E}$ **do**
    $K_s \leftarrow \mathrm{WithinSceneInduction}(\mathcal{U}_s)$
end for
$K_{\mathrm{candidate}} \leftarrow \mathrm{CrossSceneSynthesis}(\{K_s\})$
Independently review $K_{\mathrm{candidate}}$ against the action evidence and visual assessments in $\mathcal{E}$
$\mathcal{M}_{\mathrm{global}} \leftarrow \mathrm{AcceptedRules}(K_{\mathrm{candidate}})$
return $\mathcal{M}_{\mathrm{global}}$

Independent evidence review. An independent critic reviews the candidate rules against their action evidence and independent visual assessments. A rule is marked as `accepted` only when the recorded actions and visual verification support its stated conclusion and it passes the evidence review. The review considers the support for each conclusion drawn from the archived successes, failures, uncertain outcomes, and recoveries. Accepted rules are collected into $\mathcal{M}_{\mathrm{global}}$. The complete evidence archive is retained separately; only accepted knowledge is eligible for injection during evaluation. Algorithm 1 summarizes exploration and memory consolidation across adaptation instances.

3.3 Memory-Guided Task Execution

After the memory consolidation stage, the agent combines accepted knowledge from $\mathcal{M}_{\mathrm{global}}$ with the task instruction and current observations. This shared memory provides guidance on object selection, changed states, and intermediate conditions across different tasks and variants. The agent uses current observations to apply that guidance to the objects, states, and goals of the present task, then executes through robot primitives. At evaluation time, the final global memory $\mathcal{M}_{\mathrm{global}}$ is injected into the agent's context together with the task instruction and current observations. Both the global memory and the VLA parameters remain fixed throughout evaluation.

4 Experiments

We evaluate ProactiveVLA with one reference instance per task, where the agent is given a fixed interaction budget before being evaluated, testing whether proactively acquired environment knowledge transfers beyond the conditions encountered during adaptation. Harness VLA serves as the task-refinement baseline. Both ProactiveVLA and Harness VLA receive the same reference instance. On RoboCasa365 Composite-Seen, we evaluate the adaptation strategy on longer-horizon household tasks that require multiple interactions to be coordinated within a scene. We then study three components of the adaptation process, including the role of the target task, the three-level organization of exploratory goals, and the memory-management process that consolidates exploration evidence into reusable knowledge.

Table 1. Success rates (%) on LIBERO-Pro. T denotes task perturbation and S denotes position swap. Overall averages use all eight settings except for Cap-X and RATS, whose averages use their six reported settings and are marked with †. A slash indicates an unreported baseline entry. Bold numbers identify the highest score in each column.

Method	Spat-T	Spat-S	Obj-T	Obj-S	Goal-T	Goal-S	L10-T	L10-S	Overall
OpenVLA (Kim et al., 2025)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
π_0 (Black et al., 2025b)	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.3
$\pi_{0.5}$ (Black et al., 2025a)	1.0	20.0	1.0	17.0	2.0	38.0	1.0	8.0	11.0
MolmoAct (Lee et al., 2025)	0.0	0.0	0.0	6.0	0.0	0.0	6.0	0.0	1.5
NORA (Hung et al., 2025)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
X-VLA (Zheng et al., 2026)	0.0	0.0	8.0	2.0	9.0	1.0	10.0	0.0	3.8
AtomVLA (Sun et al., 2026)	1.0	16.0	0.0	10.0	11.0	2.0	9.0	1.0	6.3
Cap-X (Fu et al., 2026)	14.0	12.0	18.0	22.0	17.0	26.0	/	/	18.2†
RATS (Zhang et al., 2026a)	31.0	29.0	63.0	61.0	36.0	43.0	/	/	43.8†
π_{RLinf} (Yu et al., 2026)	42.0	59.0	71.0	78.0	45.0	42.0	49.0	14.0	50.0
Harness VLA (Zhang et al., 2026b)	81.0	69.0	94.0	91.0	75.0	66.0	**52.0**	49.0	72.1
ProactiveVLA	**85.0**	**75.0**	**98.0**	**92.0**	**88.0**	**71.0**	44.0	**58.0**	**76.4**

4.1 Experimental Setup

LIBERO-Pro. LIBERO-Pro introduces controlled perturbations to the LIBERO manipulation benchmark (Liu et al., 2023a; Zhou et al., 2026). We evaluate Spatial, Object, Goal, and LIBERO-10 under instruction redirection (T) and position swap (S). Instruction redirection changes the requested target object or goal condition, while position swap changes object placement with the instruction held fixed. Following the evaluation configuration of Harness VLA (Zhang et al., 2026b), each of the eight settings contains 10 tasks, and each task is evaluated over 10 rollouts, one for each evaluation seed. The baselines use their publicly reported results. The LIBERO-Pro overall score is the unweighted mean over its eight settings. Cap-X and RATS report only the six non-LIBERO-10 settings, and their marked averages use those reported entries.

RoboCasa365 Composite-Seen. We additionally evaluate the 16 Composite-Seen tasks from RoboCasa365 (Nasiriany et al., 2026). These tasks span multi-step kitchen activities of varying horizon, ranging from short compositions to routines involving many sequential interactions and intermediate states. All 16 task classes are represented in the benchmark's pretraining data, while evaluation is performed on target instances. For both Harness VLA and ProactiveVLA, each task is evaluated over five rollouts, one for each evaluation seed, with a budget of 30 agent turns per rollout. This setting therefore examines whether the acquired memory remains useful for longer-horizon household tasks when the agent must act under a strict evaluation-time interaction budget.

Implementation and comparisons. We instantiate ProactiveVLA on the Harness VLA execution stack with Codex as the high-level agent (Zhang et al., 2026b). In this instantiation, ProactiveVLA uses the remaining interaction budget after task success for scene-related self-proposed goals, organizes exploration around object affordances, state-changing interactions, and interaction compositions, and verifies exploratory outcomes with an independent visual checker. The resulting successful, failed, uncertain, and recovery trajectories are processed through lossless archival, experience atomization, within-scene induction, cross-scene synthesis, independent evidence review, and memory freezing. Harness VLA is evaluated with the same Codex planner, providing a matched task-refinement baseline.

4.2 Overall Results

ProactiveVLA achieves the highest overall success among the methods in Table 1, improving over Harness VLA (Codex) in seven of the eight settings. This supports the value of using interaction to acquire and consolidate knowledge beyond the requirements of the provided task.

Table 2. Per-task success rates (%) on the 16 RoboCasa365 Composite-Seen tasks. GR00T N1.5 (Bjorck et al., 2025) is evaluated with 30 rollouts per task, while the StarVLA (StarVLA Community, 2026) variants use 50 rollouts per task. Harness VLA (Zhang et al., 2026b) and ProactiveVLA are evaluated on 5 evaluation rollouts per task, one for each evaluation seed.

Task	GR00T N1.5	StarVLA-OFT	StarVLA-PI	StarVLA-GR00T	Harness VLA	ProactiveVLA
DeliverStraw	3.0	0.0	4.0	0.0	**20.0**	**20.0**
GetToastedBread	0.0	4.0	**56.0**	52.0	20.0	20.0
KettleBoiling	53.0	28.0	54.0	**60.0**	20.0	20.0
LoadDishwasher	47.0	30.0	**64.0**	42.0	60.0	40.0
PackIdenticalLunches	17.0	2.0	0.0	0.0	**20.0**	**20.0**
PreSoakPan	**70.0**	28.0	48.0	58.0	60.0	60.0
PrepareCoffee	13.0	2.0	**28.0**	**28.0**	0.0	0.0
RinseSinkBasin	60.0	28.0	76.0	**80.0**	20.0	40.0
ScrubCuttingBoard	70.0	8.0	54.0	18.0	**80.0**	**80.0**
SearingMeat	**27.0**	0.0	12.0	10.0	0.0	0.0
SetUpCuttingStation	33.0	10.0	30.0	**40.0**	20.0	**40.0**
StackBowlsCabinet	83.0	64.0	60.0	72.0	**100.0**	**100.0**
SteamInMicrowave	30.0	8.0	26.0	12.0	20.0	**60.0**
StirVegetables	33.0	0.0	10.0	24.0	0.0	**60.0**
StoreLeftoversInBowl	**43.0**	18.0	40.0	4.0	0.0	0.0
WashLettuce	67.0	20.0	68.0	66.0	40.0	**100.0**
Average	40.6	15.6	39.4	35.4	30.0	**41.3**

The largest improvement occurs on Goal-T, where execution must satisfy a redirected objective. All four position-swap settings also improve, showing that the benefit extends to altered spatial arrangements with the instruction held fixed.

On RoboCasa365 Composite-Seen, ProactiveVLA outperforms Harness VLA, as shown in Table 2. The gains are concentrated in several multi-step tasks, including `RinseSinkBasin`, `SetUpCuttingStation`, `SteamInMicrowave`, `StirVegetables`, and `WashLettuce`. These tasks require the agent to coordinate multiple interactions and intermediate states within a limited number of decisions. The result suggests that proactively acquired environment knowledge can make the available evaluation budget more useful by providing reusable information about object roles, state changes, and interaction sequences before evaluation begins.

4.3 Ablations

Table 3 isolates three components of ProactiveVLA on LIBERO-Pro Goal-S. **w/o initial task guidance** skips the environment-provided task and allocates the adaptation budget directly to autonomous exploration. **w/o memory management** keeps the collected interaction experience but bypasses the three-level memory consolidation, using the raw exploration memory for execution. **w/o three-level exploration** retains the same interaction budget and memory-management pipeline, but removes the three-level constraint on goal generation and instead allows free-form scene exploration.

Removing initial task guidance causes only a mild decrease, suggesting that autonomous exploration alone can already acquire useful environment experience, while the provided task gives exploration an additional task-relevant starting point. Removing memory management produces a similarly moderate reduction, indicating that organizing raw interaction records into reusable memory improves how the acquired experience can be used during task execution.

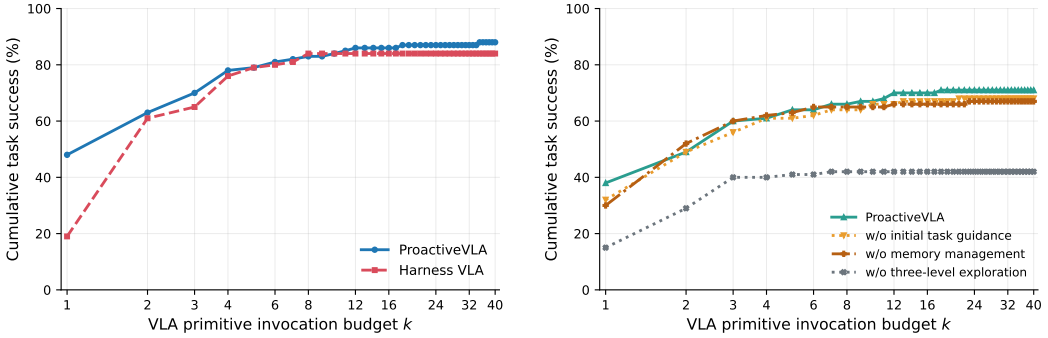
The largest degradation occurs without three-level exploration. This result indicates that the content of exploration matters more than simply increasing the amount of interaction. Free-form exploration still provides environment experience, but organizing goals around object affordances, state-changing interactions, and interaction compositions produces substantially more useful knowledge for later task execution.

5 Analysis

Proactive exploration improves execution efficiency under limited VLA calls. Figure 2 reports the fraction of evaluation instances completed with at most k VLA primitive invocations. Figure 2(a)

Table 3. Task-level success rates (%) on the LIBERO-Pro Goal-S setting. Task IDs and their corresponding instructions are listed in Appendix A.

Task	ProactiveVLA	w/o initial task guidance	w/o memory management	w/o three-level exploration
task00	0.0	0.0	0.0	0.0
task01	90.0	90.0	80.0	30.0
task02	70.0	80.0	80.0	50.0
task03	80.0	70.0	80.0	10.0
task04	100.0	100.0	100.0	90.0
task05	20.0	0.0	10.0	0.0
task06	70.0	70.0	80.0	30.0
task07	100.0	100.0	100.0	100.0
task08	100.0	100.0	90.0	70.0
task09	80.0	70.0	50.0	40.0
Overall	**71.0**	68.0	67.0	42.0



(a) Goal-T comparison

(b) Goal-S component ablations

Figure 2. Task success as a function of the maximum number of VLA primitive invocations allowed during evaluation. **(a)** ProactiveVLA has its largest advantage over Harness VLA when only a few VLA calls are available. **(b)** Structured three-level exploration contributes most strongly to this low-budget execution capability.

reports our Goal-T re-evaluation for the VLA-invocation analysis. ProactiveVLA completes 48% of instances with at most one VLA call, compared with 19% for Harness VLA. The strongest separation therefore appears at the smallest call budget, where additional controller calls are unavailable to compensate for a less effective execution strategy. This result supports the value of proactively acquired experience for completing tasks with limited use of the frozen controller, beyond improving eventual success alone. Here, k counts VLA calls during evaluation, while perception and analytic control tools remain available before and after each call. Figure 2(b) further shows that this behavior depends on how the experience is acquired and organized. The much larger drop without three-level exploration shows that free interaction alone is insufficient. Organizing exploration around object affordances, state-changing interactions, and interaction compositions is important for turning a limited adaptation opportunity into knowledge that remains useful during task execution.

Exploration-derived recovery rules guide execution when conditions change. Proactive exploration also produces conditional rules that can be reused when execution departs from the nominal trajectory. Figure 3 illustrates this behavior under target occlusion. After the manipulator blocks the object required for localization, both agents first move the arm away. ProactiveVLA then re-localizes the target before resuming execution and successfully completes the task. Harness VLA skips the re-localization step, continues operating at an incorrect location, and eventually fails. The global memory acquired by ProactiveVLA contains the corresponding rule that when the manipulator occludes an object needed for localization, the agent should first move away and then localize the object again before continuing.

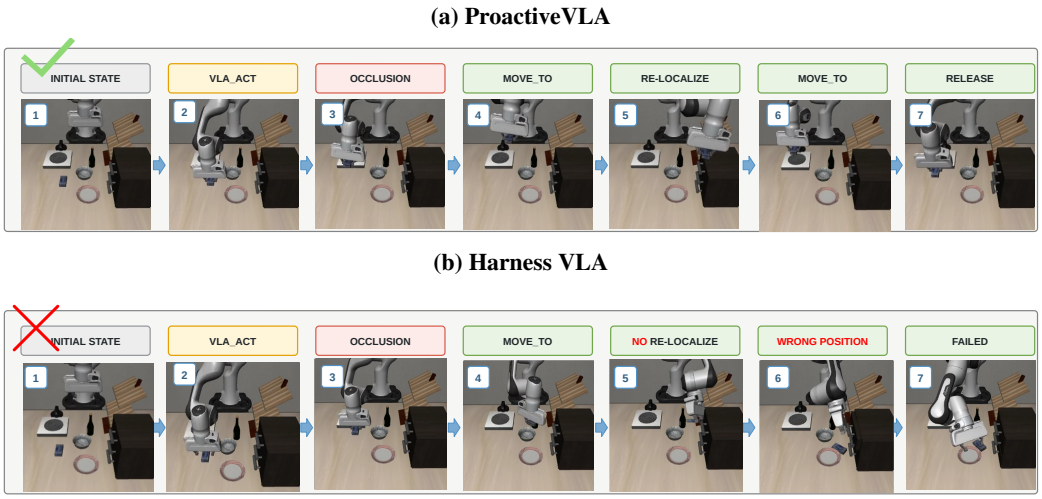


Figure 3. Execution under target occlusion. Both agents move the arm after the target becomes occluded. ProactiveVLA re-localizes before resuming the task and succeeds, while Harness VLA skips re-localization, acts at the wrong position, and fails.

Proactive exploration leads to finer-grained use of the VLA controller. Harness VLA more often gives the VLA task-level instructions such as "open the bottom drawer of the cabinet," whereas ProactiveVLA more often uses action-level instructions that specify the contact target, motion direction, and stopping condition, such as "grasp the lowest drawer's gray handle, then pull it straight outward until the bottom drawer is clearly open." Action-level instructions account for 63.4% of ProactiveVLA's VLA calls versus 34.7% for Harness VLA, while task-level instructions account for only 0.6% versus 27.3%. The difference is also reflected in tool use. After a VLA call, ProactiveVLA transitions to `move_to` in 24.7% of cases, compared with 8.4% for Harness VLA, while immediate VLA-to-VLA continuations are less frequent. These results indicate that proactive exploration helps the agent decompose execution into more specific operations and coordinate the VLA with other control tools.

6 Limitations

Our experiments are conducted in simulated manipulation environments, and the extent to which the acquired memory transfers to real-world perception, dynamics, and execution noise remains to be evaluated. Although ProactiveVLA is formulated around a general agent–primitive interface, our current implementation is instantiated on a single agentic VLA execution stack, so its effectiveness with different planners and low-level controllers is not yet established. The evaluation also uses fixed interaction budgets; different amounts or distributions of adaptation experience may change the relative benefit of proactive exploration and task refinement. Future work should evaluate the same experience-acquisition strategy across real robots, alternative system architectures, and a wider range of adaptation budgets.

7 Conclusion

We study how agentic VLA systems can use limited interaction with a new environment to acquire reusable knowledge for rapid adaptation. ProactiveVLA uses the remaining interaction budget after task completion for structured proactive exploration and consolidates successful, failed, and recovery experience into a global memory while keeping the VLA frozen. Experiments on LIBERO-Pro and RoboCasa365 Composite-Seen show that this strategy improves task execution over continued task refinement, with particularly benefits when only a few VLA calls are available. These results suggest that effective adaptation depends not only on accumulating experience, but on actively acquiring and organizing environment knowledge that can guide subsequent behavior.

AI USE STATEMENT

Generative AI tools were used throughout several stages of this work. During research ideation and execution, we used AI tools to brainstorm and refine research ideas, assist with experimental design and implementation, and support the analysis of intermediate experimental results. We also used generative AI to produce synthetic data used in our experiments, including the exploration task and memory generation described in the paper. For literature research, AI tools were used to assist with retrieval and discovery of relevant work and to summarize candidate literature during the search process.

During manuscript preparation, generative AI tools were used both to draft portions of the paper and to revise existing text for clarity, organization, grammar, and readability. The authors reviewed, revised, and verified all AI-assisted text, experimental content, references, and generated data before inclusion in the final manuscript. The authors take full responsibility for the content, claims, results, and conclusions of this work.

REFERENCES

Johan Bjorck, Valts Blukis, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Xiaowei Jiang, Kaushil Kundalia, Jan Kautz, Zhiqi Li, Kevin Lin, Zongyu Lin, Loic Magne, Yunze Man, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Jing Wang, Qi Wang, Shihao Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Seonghyeon Ye, Zhiding Yu, Yizhou Zhao, Zhe Zhang, Ruijie Zheng, and Yuke Zhu. GR00T-N1.5: An improved open foundation model for generalist humanoid robots. https://research.nvidia.com/labs/gear/gr00t-n1_5/, 2025.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In Joseph Lim, Shuran Song, and Hae-Won Park, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 17–40. PMLR, 27–30 Sep 2025a. URL https://proceedings.mlr.press/v305/black25a.html.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert 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, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025b. doi: 10.15607/RSS.2025.XXI.010.

Xin Ding, Liang Mi, Mingzhe Huang, Zixuan Wang, Chao Zhang, Zixu Hao, Fu Chen, Xiangyu Li, Yikai Zheng, Yaoyu Guo, Weijun Wang, Kun Li, Hao Wu, Yunxin Liu, and Ting Cao. Zetta ζ: An efficient closed-loop embodied harness for self-evolving physical intelligence, 2026. URL https://arxiv.org/abs/2608.16590.

Yuqing Du, Ksenia Konyushkova, Misha Denil, Akhil Raju, Jessica Landon, Felix Hill, Nando de Freitas, and Serkan Cabi. Vision-language models as success detectors. In Sarath Chandar, Razvan Pascanu, Hanie Sedghi, and Doina Precup, editors, *Proceedings of The 2nd Conference on Lifelong Learning Agents*, volume 232 of *Proceedings of Machine Learning Research*, pages 120–136. PMLR, 22–25 Aug 2023a. URL https://proceedings.mlr.press/v232/du23b.html.

Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding pretraining in reinforcement learning with large language

models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 8657–8677. PMLR, 23–29 Jul 2023b. URL https://proceedings.mlr.press/v202/du23f.html.

Letian Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Li Fei-Fei, Guanya Shi, Jiajun Wu, S. Shankar Sastry, Yuke Zhu, Ken Goldberg, and Linxi Fan. Cap-x: A framework for benchmarking and improving coding agents for robot manipulation. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=4JRO9plGAI.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and brian ichter. Inner monologue: Embodied reasoning through planning with language models. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1769–1782. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/huang23c.html.

Chia-Yu Hung, Qi Sun, Pengfei Hong, Amir Zadeh, Chuan Li, U-Xuan Tan, Navonil Majumder, and Soujanya Poria. Nora: A small open-sourced generalist vision language action model for embodied tasks, 2025. URL https://arxiv.org/abs/2504.19854.

Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Nikhil J. Joshi, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. Do as i can, not as i say: Grounding language in robotic affordances. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 287–318. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/ichter23a.html.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, Winson Han, Wilbert Pumacay, Angelica Wu, Rose Hendrix, Karen Farley, Eli VanderBilt, Ali Farhadi, Dieter Fox, and Ranjay Krishna. Molmoact: Action reasoning models that can reason in space, 2025. URL https://arxiv.org/abs/2508.07917.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500, 2023. doi: 10.1109/ICRA48891.2023.10160591.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 44776–44791. Curran Associates, Inc., 2023a. doi: 10.52202/075280-1939. URL https://proceedings.neurips.cc/paper_files/

paper/2023/file/8c3c666820ea055a77726d66fc7d447f-Paper-Datasets_and_Benchmarks.pdf.

Zeyi Liu, Arpit Bahety, and Shuran Song. Reflect: Summarizing robot experiences for failure explanation and correction. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 3468–3484. PMLR, 06–09 Nov 2023b. URL https://proceedings.mlr.press/v229/liu23g.html.

Runyu Lu, Yubo Wu, Ethan Kou, Letian Fu, Wenli Xiao, Ajay Mandlekar, Yinzhen Xu, Guanya Shi, Ken Goldberg, Ang Chen, Mosharaf Chowdhury, Yuke Zhu, Linxi "Jim" Fan, and Guanzhi Wang. ASPIRE: Agentic /Skills discovery for robotics. *arXiv preprint arXiv:2607.00272*, 2026. URL https://arxiv.org/abs/2607.00272.

Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations (ICLR)*, 2026.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jL7fwchScm.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi Jim Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick Tree Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy,

Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, and Zipeng Lin. Open x-embodiment: Robotic learning datasets and rt-x models : Open x-embodiment collaboration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903, 2024. doi: 10.1109/ICRA57147.2024.10611477.

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8583–8592. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/sekar20a.html.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530, 2023. doi: 10.1109/ICRA48891.2023.10161317.

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

Xiaoquan Sun, Zetian Xu, Chen Cao, Zonghe Liu, Yihan Sun, Jingrui Pang, Ruijian Zhang, Zhen Yang, Kang Pang, Dingxin He, Mingqi Yuan, and Jiayu Chen. Atomvla: Scalable post-training for robotic manipulation via predictive latent world models, 2026. URL https://arxiv.org/abs/2603.08519.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=ehfRiF0R3a.

Wenli Xiao, Jia Xie, Tonghe Zhang, Haotian Lin, Letian "Max" Fu, Haoru Xue, Jalen Lu, Yi Yang, Cunxi Dai, Zi Wang, Jimmy Wu, Guanzhi Wang, S. Shankar Sastry, Ken Goldberg, Linxi "Jim" Fan, Yuke Zhu, and Guanya Shi. Enpire: Agentic robot policy self-improvement in the real world, 2026. URL https://arxiv.org/abs/2606.19980.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Chao Yu, Yuanqing Wang, Zhen Guo, Hao Lin, Si Xu, Hongzhi Zang, Quanlu Zhang, Yongji Wu, Chunyang Zhu, Junhao Hu, Zixiao Huang, Mingjie Wei, Yuqing Xie, Ke Yang, Bo Dai, Zhexuan Xu, Jiakun Du, Xiangyuan Wang, Xu Fu, Letong Shi, Zhihao Liu, Kang Chen, Weilin Liu, Gang Liu, Boxun Li, Jianlei Yang, Zhi Yang, Guohao Dai, and Yu Wang. RLinf: Flexible and efficient Large-Scale reinforcement learning via Macro-to-Micro flow transformation. In *20th USENIX Symposium on Operating Systems Design and Implementation (OSDI 26)*, pages 829–846, Seattle, WA, July 2026. USENIX Association. ISBN 978-1-939133-55-7. URL https://www.usenix.org/conference/osdi26/presentation/yu-chao.

Junyi Zhang, Jiaxin Ge, Hanjun Yoo, Letian Fu, Zihan Yang, Yaowei Liu, Raj Saravanan, Shaofeng Yin, Justin Yu, Dantong Niu, Zirui Wang, Roei Herzig, Ken Goldberg, Yutong Bai, David M. Chan, Ion Stoica, Angjoo Kanazawa, Jiahui Lei, Haiwen Feng, and Trevor Darrell. Playful agentic robot learning, 2026a. URL https://arxiv.org/abs/2606.19419.

Yixian Zhang, Huanming Zhang, Feng Gao, Xiao Li, Zhihao Liu, Yi Nie, Chunyang Zhu, Jiaxing Qiu, Yuchen Yan, Jiyuan Liu, Wenhao Tang, Jiaji Rao, Zhengru Fang, Changxu Wei, Yu Wang, Wenbo Ding, and Chao Yu. Harness vla: Steering frozen vlas into reliable manipulation primitives via memory-guided agents, 2026b. URL https://arxiv.org/abs/2607.08448.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, Tai Wang, Ya-Qin Zhang, Jingjing Liu, and Xianyuan Zhan. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust, editors, *International Conference on Learning Representations*, volume 2026, pages 60580–60606, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/630d293833e09e1ecd892a898a20b074-Paper-Conference.pdf.

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, 2026. URL https://arxiv.org/abs/2510.03827.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

A LIBERO-PRO GOAL-S TASKS

Table 4 lists the 10 tasks used in the LIBERO-Pro Goal-S evaluation and their correspondence to the task IDs reported in Table 3.

Table 4. Tasks in the LIBERO-Pro Goal-S evaluation.

Task ID	Task instruction
task00	Open the middle drawer of the cabinet
task01	Put the bowl on the stove
task02	Put the wine bottle on top of the cabinet
task03	Open the top drawer and put the bowl inside
task04	Put the bowl on top of the cabinet
task05	Push the plate to the front of the stove
task06	Put the cream cheese in the bowl
task07	Turn on the stove
task08	Put the bowl on the plate
task09	Put the wine bottle on the rack

B PROMPTS USED BY PROACTIVEVLA

B.1 PROACTIVE ENVIRONMENT EXPLORATION

Provided-task acquisition prompt.

> During the task-guided acquisition stage, focus only on the provided task: `{task_language}`.
>
> Prioritize completing the task and extracting transferable interaction patterns. The environment's official success signal is the only success criterion for the provided task. Do not autonomously propose additional tasks during this stage.
>
> Budget: at most `{max_turns}` model requests and `{max_actions}` environment actions.

Proactive exploration prompt.

> You are not given an external task and must not read, infer, or reconstruct hidden benchmark goals. Your objective is to continue understanding the same scene and acquire high-quality knowledge that can transfer to future tasks. Proceed through the following three levels in order.
>
> **Level 1, object affordances.** Prioritize testing functional objects such as cabinets, drawers, and knobs for opening, closing, feedback, and internal contents. Then test how different objects can be grasped, moved, and returned.
>
> **Level 2, state-changing interactions.** Perform larger but reasonable state changes to understand the semantics of the scene. Autonomously propose interaction goals involving operations such as storage, placement, opening, closing, or other meaningful state changes.
>
> **Level 3, interaction compositions.** Construct longer tasks only by composing skills that have already been verified. For example, combine object placement with a subsequent fixture interaction. Distinguish support surfaces from container interiors.
>
> **Task-design principles.**
>
> 1. A Level-1 task must be atomic and contain exactly one visually verifiable subgoal, such as opening one drawer and confirming its displacement, or lifting one object and confirming that it has left its support surface. Do not combine grasping, transporting, and returning an object into a single Level-1 task.
> 2. Prefer goals whose before-and-after differences are clear, unobstructed, and directly visible from the global view. After an action, first move to an unobstructed observation pose and wait for the image to stabilize before judging the outcome.
> 3. If the evidence clearly shows that the goal was not completed, record the result as failure. Use uncertain only when the visual evidence is genuinely insufficient.

4. Generate memory from an autonomous task only after `complete_task` returns `verification.independently_verified=true`. Otherwise, continue with a simpler and more falsifiable task.
5. The success criterion for an autonomous task must rely only on task-local visual evidence or observable state change. Do not use `terminated`, `truncated`, official success, benchmark signals, or hidden task information during exploration.

Hard constraints.

1. Call `propose_task` before the first action of every autonomous task, and work on only one task at a time.
2. At the end of every task, call `complete_task` and then `add_memory_candidate`. Record failures and uncertain outcomes as well as successes.
3. Memory must use semantic or relative descriptions. Include applicability conditions, verification evidence, failure modes, and recovery strategies. Do not store historical absolute coordinates.
4. Do not reset by default. Request an audited reset only when an irreversible error would prevent further useful exploration. Ordinary failures must not trigger a reset.
5. Complete at least three Level-1 tasks, two Level-2 tasks, and one Level-3 task, and produce at least three controller-verified memory candidates before calling `finish_exploration`.
6. Self-evaluation in text is not evidence. Cite the actual action call IDs and preserve the continuous state of the same scene while observing before-and-after changes.

Total budget: at most `{max_turns}` model requests and `{max_actions}` environment actions.

Exploration retry policy.

For an autonomous task, do not abandon the task or switch to a new one while it remains unverified and both the task-level and global action budgets remain available. Reserve at least three environment-action attempts for each task.

After a failed grasp, opening, closing, or rotation attempt, continue within the same task. Use the observed failure evidence to change the approach direction, grasp point, gripper pose, or action magnitude. Do not repeat an identical strategy without new information.

An occluded view cannot be treated as either success or final failure.

Every tool result contains an explicit `ledger_call_id`. When calling `complete_task`, `evidence_calls` must reference the calls that actually executed the relevant action and the final unobstructed observation. Do not cite only segmentation, localization, or an earlier approach action.

The `ledger_call_id` is only an evidence identifier and is not an environment step. For `segment` or `back_project`, use the most recent `result.step`. If the current step is uncertain, use `step=-1` to retrieve the latest state.

Finalize a task only when it has been independently verified as successful, when its action budget is exhausted, or when recovery attempts confirm an irreversible error.

Visual verification prompt.

You are a visual evidence verifier completely isolated from the exploration planner. You do not know the benchmark goal and must not infer it. Verify only whether the autonomous task proposal and completion claim are directly supported by the provided pre-action image, relevant action results, and post-action image.

The verdict concerns whether the autonomous task itself was completed, not whether the textual claim is honest. If the claim correctly states that the task failed and the images confirm failure, the task verdict must still be `failed`. The overall verdict must agree with the subgoal verdicts. All subgoals must be `passed` for the overall verdict to be `passed`. If any subgoal is `failed`, the overall verdict is `failed`. Otherwise, use `uncertain`.

Decision rules.

1. `passed`: every claimed subgoal is directly confirmed by a clear before-and-after visual change.
2. `failed`: the images or action results directly contradict the claim, or clearly show that at least one claimed subgoal was not completed.

3. `uncertain`: evidence is missing, the view is occluded, the target cannot be identified, only robot contact is visible without a corresponding object or fixture state change, or the available evidence cannot establish the outcome.
4. Tool-returned `completed`/`success`, physical contact, or execution of a trajectory is not sufficient visual evidence of success.
5. If a task claims that a knob was turned and a stove was activated, but the evidence only shows robot contact and no visually verifiable knob or stove change, the verdict must be `failed` or `uncertain`, never `passed`.
6. For each subgoal, return the verdict, localized supporting evidence, contradictions, and confidence. Do not introduce information absent from the supplied evidence.

You must call `submit_visual_verdict` to return the structured result.

Provided-task memory distillation prompt.

The provided task has been confirmed successful by the environment's official success criterion. Distill the trajectory into one semantic memory item that can transfer across seeds. Describe object-relative relations, the action skeleton, success checks, common failure modes, and recovery strategies. Do not store world coordinates or pixel coordinates, and do not merely restate the task.

Task: `{task_language}`

Evidence calls: `{evidence}`

B.2 Memory Management

Memory-consolidation system prompt.

You are an experience induction module for a robot-learning experiment. Use only the provided seed-0 evidence. Separate observed facts from causal hypotheses. Do not use evaluation tasks, hidden rewards, or privileged simulator state. Output only semantic experience that can transfer across seeds. Do not use absolute world coordinates or pixel coordinates. Return the result through the specified function.

Within-scene three-level induction prompt.

Organize all attempts from this scene, including failures, retries, and recoveries, into three levels of candidate notes.

1. `object_affordance`: object properties, possible interactions, and object-conditioned operations.
2. `action_skill`: observation, approach, grasping, movement, placement, pushing, pulling, rotation, occlusion removal, and recovery.
3. `long_horizon`: step transitions, interference control, checkpoints, and recovery branches.

Both successes and failures may provide evidence. When the cause is uncertain, record it in `counterexamples` and reduce confidence. Do not turn a single observation into an unconditional rule. Every `evidence_refs` field must copy an `evidence_ref` that exists in the input. Do not restate the task and do not write coordinates.

Method: ProactiveVLA

Seed-0 task: `{task}`

Evidence: `{evidence}`

Cross-scene synthesis prompt.

Deduplicate and synthesize the notes from the ten scenes into three levels of cross-seed candidate knowledge. Preserve applicability conditions, failure counterexamples, and recovery methods. Increase confidence only when a statement is supported by at least two independent pieces of evidence or by a success-failure contrast. A conclusion supported by a single piece of evidence must retain an explicitly limited scope. Do not add facts that are absent from the scene notes. Do not use coordinates. All `evidence_refs` must preserve existing references and must not be invented.

Method: ProactiveVLA

Scene notes: `{scene_notes}`

Independent evidence-review prompt.

> You are an independent evidence reviewer. For each candidate, verify whether its stated conclusion is directly supported by the cited original evidence. The review asks whether the candidate is suitable as conditional experimental memory formed only from seed-0 interaction. It does not require the candidate to have already demonstrated a global causal effect on evaluation seeds.
>
> A single independently verified success or failure may support a narrowly scoped observation, checking rule, or same-class task step, but it cannot support a universal causal advantage.
>
> Use the following verdicts.
>
> **`accepted`**: the core content is directly supported by evidence, the applicability range is reasonable, single-sample limitations are explicit where needed, no coordinates are included, and no privileged information is leaked.
>
> **`hypothesis`**: the statement is potentially useful but the evidence is insufficient, or the proposed cause remains speculative.
>
> **`rejected`**: the candidate contradicts the evidence, has no valid citation, depends on absolute coordinates, or is not reusable.
>
> Do not accept a candidate simply because its wording appears plausible. `supported_evidence_refs` may contain only keys present in the supplied evidence index. `unsupported_claims` may list only claims actually present in the candidate and unsupported by evidence. If `verdict=accepted`, `unsupported_claims` must be empty.
>
> Method: ProactiveVLA
>
> Candidates: `{candidates}`
>
> Evidence index: `{evidence_index}`

Lossless memory-regeneration prompt.

> You are a memory-revision module for a robot-learning experiment. Regenerate `{layer_name}` using only the seed-0 exploration material from ProactiveVLA.
>
> The primary requirement is lossless organization rather than summary compression.
>
> 1. Do not compress the direct memory items and scene notes into a small number of generic principles.
> 2. Preserve each atomic source item by default. Merge items only when they are fully redundant, and preserve all corresponding `source_item_ids`.
> 3. Every output item must preserve executable details, conditions, observed facts, action steps, success verification, failure modes, recovery methods, counterexamples, confidence, and `evidence_refs`.
> 4. Do not remove failed or uncertain experience. When a cause is speculative, mark it explicitly as a hypothesis or place it in `counterexamples`.
> 5. The three layers are retrieval layers rather than three summaries. The first describes what objects can do and under what limitations. The second describes how actions should be executed and verified. The third describes how multi-stage tasks should be connected, checked, and recovered.
> 6. The same source item may map to multiple layers. Do not discard cross-layer information for the sake of deduplication.
> 7. Return `coverage` and list, for every `source_item_id`, which generated entries cover it. If coverage is partial, explicitly record `omitted_information`. Coverage should be complete by default.
> 8. Do not use evaluation seeds, hidden goals, rewards, `terminated`, or privileged state. Do not store historical absolute coordinates or pixel coordinates.
> 9. Call `save_lossless_layer` to return the result.
>
> Current layer: `{layer}`
>
> Direct memory: `{direct}`
>
> Scene-level notes: `{scene_items}`

B.3 EVALUATION-TIME MEMORY INJECTION

[ProactiveVLA MEMORY INJECTION]
The full frozen ProactiveVLA memory is provided below. Use it as the only memory source for this run. Do not read, modify, or create external memory files. If another instruction asks you to read an external memory file, treat the injected memory below as the authoritative memory and do not search the filesystem for additional memory.
`{frozen_memory_json}`