CARE-VLA: CONTEXT-AWARE RETRIEVAL OF EXECUTION EXPERIENCE FOR AGENTIC VLA PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic vision–language–action (VLA) systems use external planners and execution memories to support frozen control policies. However, task- or episode-level memories do not directly align with local planning decisions, and irrelevant or redundant experience can distract the planner. We propose CARE-VLA, a training-free framework for context-aware retrieval of step-level execution experience. Its read-only multimodal bank links each primitive execution to its preceding context, outcome, and before–after observations, allowing local evidence to be retrieved across tasks and episodes. At each decision boundary, CARE-VLA filters records by execution-context compatibility, fuses task-and-context and visual rankings, and selects evidence with diverse next primitives and distinct source episodes. A compact Memory Card supplements the planner's original context without updating planner or VLA parameters. On LIBERO-Pro and RoboTwin C2R, CARE-VLA achieves success rates of 80.3% and 61.6%, exceeding the published Harness VLA results with the corresponding Codex planners by 8.1 and 3.6 percentage points. On LIBERO-Pro episodes successfully completed by both methods, it reduces average primitive executions by 8.9%. These results support step-level experience retrieval for improving frozen agentic VLA planning.

1 INTRODUCTION

Vision–language–action (VLA) models map multimodal inputs directly to robot actions and have shown promising results in simulation and real-world manipulation (Zitkovich et al., 2023; Kim et al., 2025; Physical Intelligence et al., 2025). These capabilities, however, often require substantial training data and computational resources. Changes in lighting, backgrounds, object positions, and other environmental factors can create a mismatch between deployment conditions and expert demonstrations, reducing task success and generalization. Meanwhile, advances in large language and vision-language models have enabled agents to handle increasingly complex planning and decision-making tasks. This has motivated agentic VLA frameworks such as Harness VLA (Zhang et al., 2026b) and RoboHarness (Li et al., 2026). These frameworks keep the underlying VLA frozen and introduce an external agentic harness to coordinate execution and improve deployment robustness without updating the VLA parameters.

Agentic VLA systems record execution histories and feedback as reusable memory to support later planning. Yet selecting appropriate next actions remains challenging in long-horizon tasks, particularly under environmental changes or following execution failures. With the underlying VLA frozen, better organization and use of execution experience offers a way to improve these decisions. Harness VLA maintains two types of memory. For each task, task-specific memory contains an execution trace and a task summary. Global memory provides a task-independent summary of primitive usage rules and failure diagnosis (Zhang et al., 2026b). RoboHarness uses episode-level multimodal memory that stores past executions, success or failure outcomes, and diagnostic information for failure-aware retrieval and attribution (Li et al., 2026).

Two challenges remain in using such memories for planning. **(1) Decision-level misalignment.** Memories are mainly organized around complete tasks, episodes, or general execution knowledge, whereas the agent chooses its next action at a finer decision boundary. The planner must therefore identify which parts of a historical execution are relevant to its current decision, increasing its reasoning burden and potentially leading to suboptimal decisions. **(2) Inefficient experience retrieval.**

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

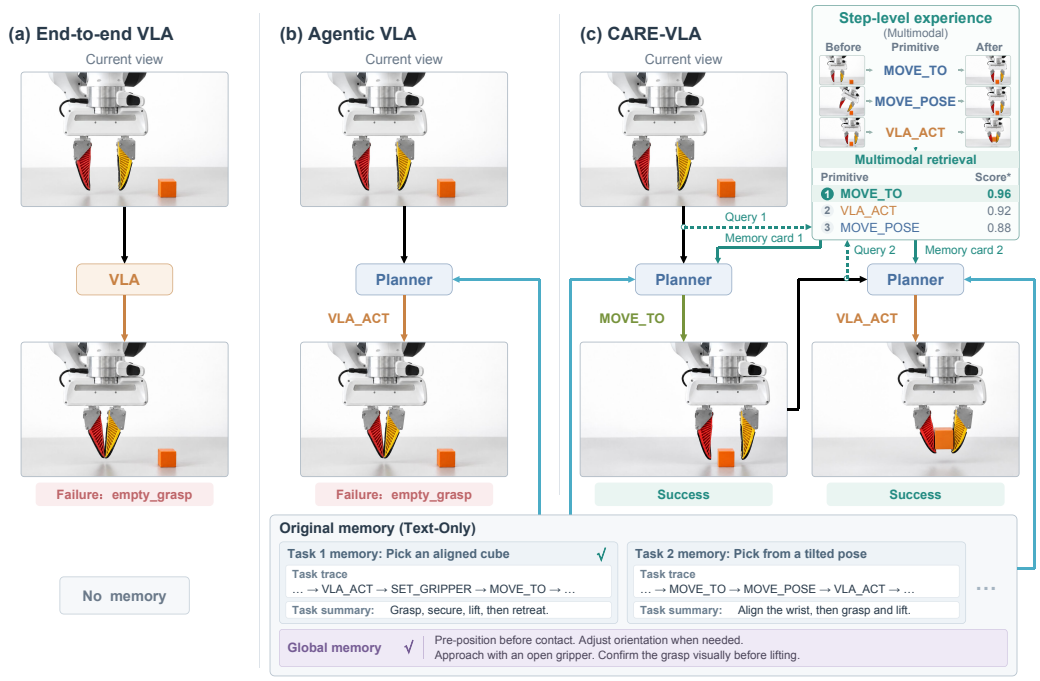


Figure 1: Schematic illustration of context-aware step-level experience retrieval for agentic VLA planning. (a) An end-to-end VLA acts directly on the current observation. (b) An agentic VLA adds a planner supported by task-specific traces and a global summary, but must infer which historical experience applies to the current decision. In the illustrated scenario, both attempt a grasp without first correcting gripper–object misalignment. (c) CARE-VLA retrieves context-compatible step-level experience using task semantics, execution context, and visual cues, and provides a Memory Card before each planning decision. The planner first selects MOVE_TO to align the gripper, then queries again using the updated observation and feedback before invoking VLA_ACT for grasping.

As memory accumulates, identifying experience that is both relevant to the current decision and compatible with its execution context becomes difficult. Without suitable filtering and selection, irrelevant, context-mismatched, or redundant records can distract the planner rather than help it.

We propose **CARE-VLA** (Context-Aware Retrieval of Execution Experience for Agentic VLA), a training-free framework for retrieving step-level multimodal evidence. Splitting reference trajectories at primitive decision boundaries makes local experience independently accessible across tasks and episodes. A shared bank can therefore pool broader execution experience without requiring the planner to follow an entire historical trajectory. At each decision, context-compatibility filtering, multimodal ranking, and primitive-diverse selection produce a compact Memory Card. The bank can grow while the number of records presented to the planner remains bounded, allowing broader experience reuse without filling the context with repeated action examples. Our main contributions are:

1. We organize execution context, actions, outcomes, and before–after observations into step-level multimodal records aligned with primitive decision boundaries, enabling local experience reuse across tasks and episodes.
2. We develop a context-aware multimodal retrieval mechanism that combines execution-context filtering, semantic and visual rank fusion, and diversity-based selection to provide compact, relevant, and complementary historical evidence for planning.
3. We evaluate CARE-VLA on LIBERO-Pro (Zhou et al., 2025) and RoboTwin C2R. Without updating planner or VLA parameters, it achieves success rates of 80.3% and 61.6%, respectively, exceeding the published Harness VLA results with the corresponding Codex planners by 8.1 and 3.6 percentage points.

2 Related Work

2.1 VLA Models and Agentic Planning

Vision–language–action (VLA) models map visual observations and language instructions to robot actions. RT-2 transfers vision-language knowledge to robot control through joint training on web data and robot trajectories (Zitkovich et al., 2023). OpenVLA learns manipulation policies from diverse demonstrations (Kim et al., 2025), while $\pi_{0.5}$ uses heterogeneous data and tasks to improve generalization to new environments (Physical Intelligence et al., 2025).

Language-model-based planners compose robot capabilities and incorporate execution feedback. Code as Policies generates programs using perception and control APIs (Liang et al., 2023), while Inner Monologue uses environmental feedback for closed-loop planning (Huang et al., 2023). Harness VLA extends this setup to a frozen VLA, coordinating it as a retryable primitive with analytic primitives (Zhang et al., 2026b). RoboHarness uses experience-based diagnosis and tool interventions to adjust a frozen VLA's inputs and execution flow (Li et al., 2026). We follow this external planning setup and study how historical experience can support planner decisions.

2.2 Memory Organization for Embodied Decision-Making

Memory supports decisions that require information beyond the current observation. MemoryVLA retrieves perceptual and cognitive representations of past observations for action generation (Shi et al., 2026). MemER learns to select and track rollout keyframes, combining them with recent observations to instruct a low-level policy (Sridhar et al., 2026). Both learn to use history within a rollout.

For cross-episode reuse, Harness VLA stores an execution trace and a task summary for each task, alongside a task-independent summary of primitive usage rules and failure diagnosis (Zhang et al., 2026b). RoboHarness stores multimodal experiences and refines diagnostic annotations through offline consolidation (Li et al., 2026). Such task-level organization still requires identifying which local experience applies after perturbations or failures. We therefore organize trajectories into step-level multimodal records aligned with primitive decision boundaries. Each links the pre-decision context to the historical next primitive, its outcome, and before–after observations, enabling independent retrieval across episodes.

2.3 Experience Retrieval and In-Context Adaptation

Demonstrations and retrieved experience support inference-time adaptation. ICRT learns from contextual demonstrations through autoregressive trajectory modeling (Fu et al., 2024); RICL post-trains a VLA for in-context learning with retrieved demonstration segments (Sridhar et al., 2025). MAP-VLA retrieves learned stage-level soft prompts (Li et al., 2025), while Dejavu uses retrieved step-level experience to condition a trained residual controller around a frozen VLA (Wu et al., 2026). Our approach instead supplies primitive-level evidence to an external planner without training an additional controller or memory representation.

For external planners, Harness VLA provides task-matched structures and global rules, while RoboHarness retrieves experiences through multimodal similarity (Zhang et al., 2026b; Li et al., 2026). However, similar tasks or scenes may differ in local execution state, and retrieved records may repeat the same primitive choice. We combine execution-context filtering, multimodal ranking fusion, primitive diversity, and source-episode deduplication. Retrieved actions and outcomes form a Memory Card for comparison with current observations. This requires no additional training of the planner, VLA, or memory prompts, while leaving action selection and argument grounding to the planner.

3 Method

Figure 2 summarizes CARE-VLA. We first review Harness VLA's planning and execution loop, then describe experience bank construction, multimodal retrieval, and integration through a Memory Card.

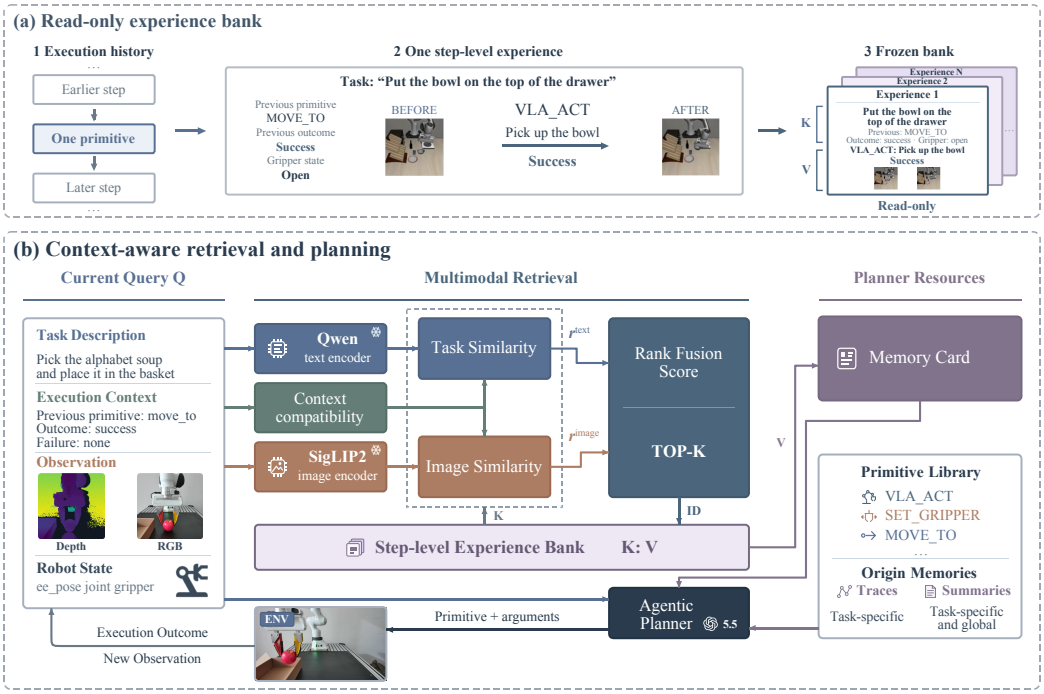


Figure 2: **Overview of CARE-VLA. (a) Step-level experience bank.** Reference execution histories are split into primitive-level transitions. Each record stores the execution context, executed primitive, its outcome, and before and after images in a read-only bank. **(b) Context-aware multimodal retrieval.** At each decision boundary, CARE-VLA filters records by execution-context compatibility, combines task-and-context and visual rankings, and applies thresholding and diversity constraints to select up to three records. The selected evidence forms a Memory Card that supplements the planner's original context. Primitive execution returns observations and feedback for the next decision.

3.1 Preliminary: Planning and Execution in Harness VLA

Harness VLA uses a high-level planner to complete tasks through executable primitives (Zhang et al., 2026b). The primitive runs through its corresponding VLA or analytic implementation and returns new observations and execution feedback. For each task, its task-specific memory contains an execution trace and a task summary. Its global memory provides a task-independent summary of primitive usage rules and failure diagnosis. Together, they provide task experience and execution knowledge for planning.

Let t denote a decision boundary after a primitive execution, ℓ the task instruction, o_t the current observation, and $\mathcal{H}_t$ the planner's original context, including the execution history and existing memories. The frozen high-level planner π_{P} selects a primitive a_t from a fixed library $\mathcal{A}$ and its executable arguments θ_t:

$$(a_t, \theta_t) = \pi_{\mathrm{P}}(\ell, o_t, \mathcal{H}_t), \qquad a_t \in \mathcal{A}. \tag{1}$$

Task traces and strategy summaries provide an overall reference, but the planner still needs to identify which historical steps are relevant to its current decision. CARE-VLA therefore introduces step-level retrieval between execution feedback and the next planning decision, providing relevant historical transitions through a Memory Card without changing the primitives or their implementations.

3.2 Step-Level Multimodal Experience Bank Construction

As shown in Figure 2(a), we split reference trajectories into primitive-level transitions to align memory with the planner's decision granularity. Each record links the pre-execution context, executed

primitive, and outcome. Retrieval operates over a shared bank across tasks and episodes, matching local execution conditions without requiring the current episode to follow an entire past action sequence.

The bank is constructed offline before evaluation and denoted by $\mathcal{M} = \{m_i\}_{i=1}^{M}$. Each step-level experience record is $m_i = (k_i, v_i, s_i)$, where k_i is the retrieval key, v_i contains historical evidence for the planner, and s_i identifies the source episode and transition. The key is

$$k_i = (\ell_i, p_i, y_i, f_i, g_i, h_i, I_i^{\text{before}}). \tag{2}$$

Here, ℓ_i is the historical task instruction; p_i, y_i, and f_i are the previous primitive, its local outcome, and failure type; g_i is the gripper state; h_i is the recent primitive history; and I_i^{before} is the agent-view image before the next action. Outcomes and failure types are taken directly from Harness VLA's primitive feedback, using the same feedback rules during bank construction and evaluation.

The value v_i contains the historical next primitive a_i, its local outcome, the source episode's final outcome, and the before–after images. The key's outcome y_i describes the situation before the historical decision, whereas the local outcome in v_i describes what happened after executing a_i.

The before image supports matching with the current scene, while the after image shows the scene change following the historical action. Source metadata retains episode and step identifiers for tracing records to their original executions and for deduplication during retrieval. Reference experience is collected with seeds excluded from evaluation. The bank remains read-only during evaluation, with no evaluation trajectories added. Historical execution arguments are not directly reused. The planner determines current targets and spatial arguments from live observations.

3.3 Context-Aware Multimodal Experience Retrieval

Figure 2(b) illustrates the retrieval process. At each decision boundary, we form a query from the task instruction, current agent-view image, and execution context:

$$q_t = (\ell, I_t, x_t), \qquad x_t = (p_t, y_t, f_t, g_t, h_t). \tag{3}$$

The query fields have the same meanings as their counterparts in the bank. Retrieval applies context filtering, two ranking routes, rank fusion and thresholding, and finally selection of a small set of experiences with different next primitives.

Execution-context filtering. Similar tasks or images do not necessarily require the same next action. For example, scenes after a successful and a failed grasp can look similar, although the next decision must account for different execution states. We retain records whose previous primitive, local outcome, and failure type match the query. Gripper states must also match when both are known. This gives the compatible set $\mathcal{C}_t$, within which both ranking routes operate.

Task-and-context ranking. Task semantics alone cannot distinguish different execution stages of the same task. We therefore combine task relevance with local execution-context matching. We first compute

$$C_i = \beta_{\text{ctx}} c_i + \beta_{\text{hist}} H_i + \beta_{\text{fail}} F_i, \tag{4}$$

where c_i measures how closely the historical execution state matches the current state, H_i compares the three most recent primitives in the query and the historical record, and F_i measures whether their failure types match. The weights are $(\beta_{\text{ctx}}, \beta_{\text{hist}}, \beta_{\text{fail}}) = (0.50, 0.30, 0.20)$.

Task semantic similarity is computed from instruction embeddings:

$$T_i = \frac{e(\ell)^\top e(\ell_i)}{\|e(\ell)\|_2 \, \|e(\ell_i)\|_2}, \tag{5}$$

where e is the frozen Qwen3-Embedding-0.6B encoder (Zhang et al., 2025). We combine task similarity and context matching as

$$U_i = \lambda T_i + (1 - \lambda) C_i, \tag{6}$$

where $\lambda = 0.15$. After compatibility filtering, c_i and F_i are constant across eligible records, so their relative order depends on task similarity and recent primitive history. Appendix A provides the matching definitions and encoder details.

Visual ranking. Text and execution feedback do not fully describe scene information such as object positions and spatial relations. The second route therefore compares the current agent-view image with the historical before image:

$$V_i = \phi(I_t)^\top \phi(I_i^{\text{before}}), \tag{7}$$

where ϕ is a frozen SigLIP2 encoder with unit-normalized outputs (Tschannen et al., 2025). Both images use the agent view, providing scene-level visual information rather than wrist-view matching. We rank records by U_i and V_i separately and take the union of the top N records from each route to form the candidate pool $\mathcal{P}_t$, with $N = 12$. Ranks are computed over each route's complete compatible list before truncation. The visual route uses only before images; historical after images do not participate in ranking.

Rank fusion and thresholding. The raw scores from the two routes have different scales, so a direct weighted sum would depend on their numerical ranges. We instead combine their ranks. Let $r_i^{\text{text}}, r_i^{\text{vis}} \in \mathbb{N}_+$ denote the task-and-context rank and the visual rank of record i, respectively. We use a normalized reciprocal-rank score (Cormack et al., 2009):

$$\begin{aligned} \rho(r) &= \frac{c+1}{c+r}, \\ S_i &= (1-\alpha)\rho(r_i^{\text{text}}) + \alpha\rho(r_i^{\text{vis}}). \end{aligned} \tag{8}$$

The constant c controls rank-score decay, and α controls the visual weight; we set $c = 10$ and $\alpha = 0.5$. Records without an available historical image use the text route alone, $S_i = \rho(r_i^{\text{text}})$.

Lower-ranked candidates need not be added to the planning context. For a nonempty pool $\mathcal{P}_t$, let $S_{\max} = \max_{m_i \in \mathcal{P}_t} S_i$. We retain records satisfying

$$\begin{aligned} S_i &\geq \tau_t, \\ \tau_t &= \max(\tau_{\text{abs}}, S_{\max} - \Delta). \end{aligned} \tag{9}$$

Here, $\tau_{\text{abs}} = 0.55$ is the minimum fused-score threshold, and $\Delta = 0.15$ is the maximum allowed gap from the best candidate. Together, they exclude low-scoring records while retaining alternatives close to the best match.

Diverse evidence selection. Taking the highest-scoring records directly can repeat the same next action or fill several card entries with similar steps from one trajectory. Here, the next-primitive type is identified by the historical primitive stored in the record. To preserve alternative actions and reduce repeated evidence, we scan eligible records in descending fused-score order and accept a record only if neither its next-primitive type nor its source episode has already been selected. Each next-primitive type and source episode therefore contributes at most one record. Selection stops at $K_{\max} = 3$ records or when candidates are exhausted, without forcing the card to be filled. If no record qualifies, no step-level evidence is provided.

3.4 Memory Card and Integration with the Planning Loop

CARE-VLA packages selected records into a Memory Card $\mathcal{E}_t$. Each entry contains the historical execution context, next primitive, and outcome, with available before–after images displayed side by side as historical observations. The card supplements Harness VLA's original context $\mathcal{H}_t$, including its task-specific and global memories:

$$\begin{aligned} \mathcal{E}_t &= \text{Retrieve}(\mathcal{M}, q_t), \\ (a_t, \theta_t) &= \pi_{\text{P}}(\ell, o_t, \mathcal{H}_t, \mathcal{E}_t). \end{aligned} \tag{10}$$

The planner grounds its decision and arguments in the current scene; empty retrieval leaves the original context unchanged. Execution follows Harness VLA's existing path, and the returned observations and feedback form the next query. Planner and VLA parameters remain frozen, and the bank remains read-only.

Table 1: **Success rates on LIBERO-Pro (%).** CARE-VLA uses 10 tasks $\times$ 10 seeds per suite, following Harness VLA. Baseline values are taken from the comparison in Zhang et al. (2026b). Bold and underlined values indicate the best and second-best results in each column, respectively. – denotes unreported results. † denotes an average over the six reported suites.

Method	Spat-T	Spat-S	Obj-T	Obj-S	Goal-T	Goal-S	Long-T	Long-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., 2025)	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.3
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	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., 2025)	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., 2025)	42.0	59.0	71.0	78.0	45.0	42.0	49.0	14.0	50.0
Harness VLA (CC) (Zhang et al., 2026b)	**94.0**	80.0	88.0	90.0	**87.0**	**87.0**	**71.0**	**62.0**	**82.4**
Harness VLA (Codex) (Zhang et al., 2026b)	81.0	69.0	<u>94.0</u>	91.0	75.0	66.0	52.0	49.0	72.1
CARE-VLA (Codex, text-only)	84.0	<u>88.0</u>	**96.0**	<u>94.0</u>	78.0	<u>79.0</u>	59.0	46.0	78.0
CARE-VLA (Codex)	<u>92.0</u>	**90.0**	93.0	**95.0**	<u>80.0</u>	75.0	<u>62.0</u>	<u>55.0</u>	<u>80.3</u>

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluate CARE-VLA on LIBERO-Pro (Zhou et al., 2025) and RoboTwin C2R. LIBERO-Pro contains eight suites, each with 10 tasks evaluated over 10 seeds per task, giving 800 episodes in total. For RoboTwin C2R, we follow the Harness VLA protocol with 50 tasks and 5 randomized seeds per task, giving 250 episodes. Reference experience is collected with seeds excluded from evaluation, and the experience bank remains read-only.

Implementation. Experiments run on eight NVIDIA RTX 4090 GPUs. CARE-VLA uses GPT-5.5 through Codex CLI, with medium reasoning on LIBERO-Pro and xhigh reasoning on RoboTwin C2R. These settings match the corresponding Harness VLA (Codex) configurations. Harness VLA (CC), where CC denotes Claude Code, uses a different planner and is included as an additional system-level comparison. CARE-VLA retains Harness VLA's primitive implementations, feedback rules, execution budgets, and original task-specific and global memories. The frozen VLA backends are π_{RLinf} on LIBERO-Pro and LingBot-VLA on RoboTwin C2R. The planner and VLA parameters remain frozen. Task instructions and images are encoded with frozen `Qwen/Qwen3-Embedding-0.6B` and SigLIP2 encoders, respectively.

Experience bank. The LIBERO-Pro bank contains 1,093 step-level records extracted from 80 reference episodes across eight suites (Appendix B). The RoboTwin bank pools `demo_clean` reference executions from all 50 tasks, using one non-evaluation initialization seed per task. Both banks support retrieval across tasks and episodes. For the experience-source study, the 1-, 4-, and 8-suite banks use the target suite, Spat-T/S and Goal-T/S, and all eight LIBERO-Pro suites, respectively.

4.2 Simulation Results

LIBERO-Pro. With the same Codex planner configuration, CARE-VLA achieves 80.3% overall success, compared with the 72.1% of Harness VLA (Codex) (Table 1). This improvement narrows the gap to Harness VLA (CC), which achieves 82.4% with a different planner. The results support step-level experience retrieval as a way to improve a fixed agentic VLA system without updating planner or VLA parameters.

RoboTwin C2R. With the same frozen LingBot-VLA backend, CARE-VLA achieves 61.6% success, exceeding the reported Harness VLA (Codex) and Harness VLA (CC) by 3.6 and 3.2 percentage points, respectively (Table 2). These gains support experience reuse within the same agentic execution framework on a second benchmark, without additional planner or VLA training.

Table 2: **Success rates on RoboTwin C2R (%).** Q and H denote results reported by Qwen Team (2026) (Hard setting) and Zhang et al. (2026b), respectively. CARE-VLA uses 50 tasks × 5 randomized seeds (250 episodes), following Harness VLA; external rows retain their source protocols. Bold marks the best reported result.

Method	Source	Success rate
GR00T-N1.7	Q	20.7
$\pi_{0.5}$	Q	47.9
StarVLA	Q	10.6
Qwen-RobotManip	Q	62.6
Qwen-RobotManip-Context	Q	**69.4**
LingBot-VLA	H	50.4
Harness VLA (Codex)	H	58.0
Harness VLA (CC)	H	58.4
CARE-VLA (Codex)	This work	61.6

Table 3: **Experience sources and primitive diversity.** Success rates (%) on Spat-T/S and Goal-T/S. Overall is the average across these four suites. All methods use the Codex planner. All CARE-VLA settings retain source-episode deduplication and select at most three records. Only the w/o diversity variant removes next-primitive diversity.

Method / Setting	Spat-T	Spat-S	Goal-T	Goal-S	Overall
Harness VLA (Codex)	81.0	69.0	75.0	66.0	72.8
CARE-VLA: 1-suite bank	87.0	81.0	**81.0**	70.0	79.8
CARE-VLA: 4-suite bank	91.0	87.0	77.0	72.0	81.8
CARE-VLA: 8-suite bank, w/o diversity	84.0	73.0	72.0	61.0	72.5
CARE-VLA: 8-suite bank, full	**92.0**	**90.0**	80.0	**75.0**	**84.3**

4.3 Ablation Studies

All ablations use the same task–seed pairs, planner, thresholds, and remaining settings, changing only the factor under study.

Visual information. CARE-VLA (text-only) removes visual ranking ($\alpha = 0$) and historical Card images, retaining only the top N task-and-context candidates. The bank, thresholds, diversity constraints, and live planner observations remain unchanged. Success drops from 80.3% to 78.0% (Table 1), supporting the combined value of visual retrieval and historical images; their individual contributions are not separately isolated.

Broader experience reuse. Step-level records allow local evidence to be pooled across tasks and episodes. We evaluate this use case by expanding the bank from the target suite to Spat-T/S and Goal-T/S, then to all eight suites, while retaining the same three-record Card limit and retrieval settings. Average success rises from 79.8% to 81.8% and 84.3% (Table 3). This trend shows that broader and larger experience banks can improve planning under a fixed Card limit. The study evaluates the joint benefit of expanded coverage and experience volume.

Primitive diversity. Removing next-primitive diversity lowers average success from 84.3% to 72.5%, while retaining source-episode deduplication and a budget of at most three records (Table 3). This supports selecting diverse next-primitive evidence rather than relying solely on retrieval scores.

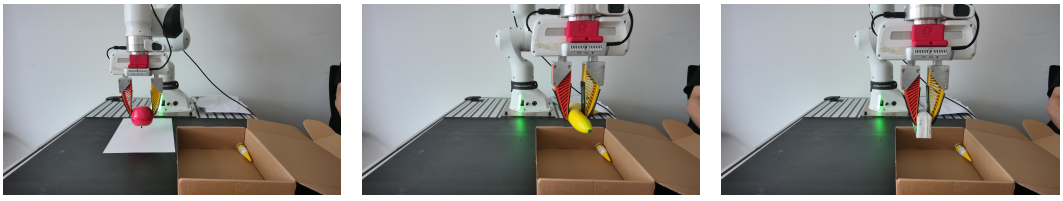
4.4 Efficiency

Historical runs record a decision-weighted mean retrieval-core latency of 93.2 ms over 10,553 decisions, including visual encoder requests. This documents roughly 0.1 s per decision in those configurations. CARE-VLA reduces average primitive executions by 8.9% on the 548 LIBERO-Pro

task–seed pairs solved by both methods. Appendix C gives historical configurations and timing exclusions.

Table 4: **Real-robot results.** Task columns report successful trials out of 20 evaluations. Overall reports the average success rate across the three tasks.

Method	Apple	Banana	Milk	Overall
Fine-tuned $\pi_{0.5}$	13/20	10/20	9/20	53.3%
Harness VLA (Codex)	16/20	14/20	14/20	73.3%
CARE-VLA (Codex)	19/20	17/20	15/20	85.0%



(a) Pick and place apple (b) Pick and place banana (c) Pick and place milk

Figure 3: **Real-robot tasks.** Three pick-and-place tasks evaluated on a Franka Research 3 robotic arm.

4.5 Real-Robot Experiments

Setup and tasks. We evaluate on a Franka Research 3 robotic arm using three pick-and-place tasks involving an apple, a banana, and a milk container, as shown in Figure 3. Observations consist of third-person RGB and depth images, without wrist-camera inputs. For each task, we collect 30 demonstration trajectories and fine-tune a separate $\pi_{0.5}$ policy. Each method is evaluated over 20 trials per task.

Compared methods. The fine-tuned $\pi_{0.5}$ baseline, Harness VLA, and CARE-VLA share the same task-specific policy. Both agentic methods use GPT-5.5 through Codex CLI with xhigh reasoning. Harness VLA uses one of CARE-VLA's five reference episodes per task to form its task-specific memory and derives global memory from these selected executions. CARE-VLA retains these memories and constructs its step-level bank from all 15 reference episodes (Appendix D).

Results. CARE-VLA achieves 85.0% success, compared with 73.3% for Harness VLA and 53.3% for the fine-tuned $\pi_{0.5}$ policy. The 11.7-percentage-point gain over Harness VLA supports augmenting the original memories with step-level evidence from the broader reference bank in these real-robot tasks.

5 Conclusion

CARE-VLA makes step-level execution experience reusable across tasks and episodes by supplying context-compatible, primitive-diverse evidence at each planning decision. Simulation and real-robot results support its ability to improve a given agentic VLA system without additional parameter training. Expanding the shared bank improves performance while the Card remains limited to three records. Effectiveness depends on memory coverage and retrieval quality; future work will study adaptive retrieval and reliable memory updates.

AI Use Statement

We used generative AI tools to polish the writing and assist in drafting portions of the manuscript based on content provided by the authors. All AI-assisted text was reviewed and revised by the authors, who take full responsibility for the accuracy and integrity of the final manuscript.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, et al. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025. URL https://arxiv.org/abs/2410.24164.

Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 758–759, 2009. doi: 10.1145/1571941.1572114.

Letian Fu, Huang Huang, Gaurav Datta, Lawrence Yunliang Chen, William Chung-Ho Panitch, Fangchen Liu, Hui Li, and Ken Goldberg. In-context imitation learning via next-token prediction. arXiv:2408.15980, 2024. URL https://arxiv.org/abs/2408.15980.

Letian Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, et al. CaP-X: A framework for benchmarking and improving coding agents for robot manipulation. arXiv:2603.22435, 2026.

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 *Proceedings of the Conference on Robot Learning (CoRL)*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1769–1782, 2023.

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. arXiv:2504.19854, 2025.

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 *Proceedings of the Conference on Robot Learning (CoRL)*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713, 2025.

Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, et al. MolmoAct: Action reasoning models that can reason in space. arXiv:2508.07917, 2025.

Runhao Li, Wenkai Guo, Zhenyu Wu, Changyuan Wang, Haoyuan Deng, Zhenyu Weng, Yap-Peng Tan, and Ziwei Wang. MAP-VLA: Memory-augmented prompting for vision-language-action model in robotic manipulation. arXiv:2511.09516, 2025. URL https://arxiv.org/abs/2511.09516.

Zhuoran Li, Zhiyang Li, Kaijun Zhou, and Jinyu Gu. RoboHarness: A memory-augmented policy harness for vision-language-action model robustness via in-context adaptation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2026. URL https://arxiv.org/abs/2603.24060.

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 *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9493–9500, 2023. doi: 10.1109/ICRA48891.2023.10160591.

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:2504.16054, 2025.

Qwen Team. Qwen-RobotManip Technical Report: Alignment Unlocks Scale for Robotic Manipulation Foundation Models, 2026. URL https://arxiv.org/abs/2606.17846.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. MemoryVLA: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2508.19236.

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. MemER: Scaling up memory for robot control via experience retrieval. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2510.20328.

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

Xiaoquan Sun, Zetian Xu, Chen Cao, Zonghe Liu, Yihan Sun, Jingrui Pang, et al. AtomVLA: Scalable post-training for robotic manipulation via predictive latent world models. arXiv:2603.08519, 2026.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, et al. SigLIP 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. arXiv:2502.14786, 2025.

Shaokai Wu, Yanbiao Ji, Qiuchang Li, Zhiyi Zhang, Qichen He, Wenyuan Xie, Guodong Zhang, Bayram Bayramli, Yue Ding, and Hongtao Lu. Dejavu: Towards experience feedback learning for embodied intelligence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 29578–29587, 2026. URL https://arxiv.org/abs/2510.10181.

Chao Yu, Yuanqing Wang, Zhen Guo, Hao Lin, Si Xu, Hongzhi Zang, Quanlu Zhang, Yongji Wu, Chunyang Zhu, Junhao Hu, et al. RLinf: Flexible and efficient large-scale reinforcement learning via macro-to-micro flow transformation. arXiv:2509.15965, 2025.

Junyi Zhang, Jiaxin Ge, Hanjun Yoo, Letian Fu, Zihan Yang, Yaowei Liu, et al. Playful agentic robot learning. arXiv:2606.19419, 2026a. URL https://arxiv.org/abs/2606.19419.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, et al. Qwen3 Embedding: Advancing text embedding and reranking through foundation models. arXiv:2506.05176, 2025.

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

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, et al. X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. arXiv:2510.10274, 2025.

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

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 *Proceedings of the Conference on Robot Learning (CoRL)*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183, 2023.

A Retrieval Implementation Details

A.1 Execution Context and Recent History Matching

CARE-VLA first filters records by execution-context compatibility. The previous primitive, previous outcome, and failure type must match those in the current query. Gripper states must also match when both are known. These constraints apply to the context before the historical action; they do not require records to recommend the same next primitive.

Let $d(u, v)$ indicate equality between two fields after normalizing case and removing leading and trailing whitespace. Let $z(y)$ indicate whether an outcome is `TERMINATED` or `TRUNCATED`. Execution-context and failure-type matching are defined as

$$c_i = \frac{d(p_t, p_i) + d(y_t, y_i) + \mathbf{1}[z(y_t) = z(y_i)]}{3}, \qquad F_i = d(f_t, f_i). \tag{11}$$

Recent-history matching compares the three most recent primitives in the query and the historical record:

$$H_i = \sum_{j=1}^{3} w_j d(h_t[-j], h_i[-j]), \qquad (w_1, w_2, w_3) = (0.6, 0.3, 0.1). \tag{12}$$

Here, $h[-1]$ denotes the most recent primitive. More recent actions receive larger weights. Sequences shorter than three steps are padded with missing values. Two missing values count as a match, and the weights are not renormalized.

The context score and the combined task-and-context score are

$$\begin{aligned} C_i &= \beta_{\text{ctx}} c_i + \beta_{\text{hist}} H_i + \beta_{\text{fail}} F_i, \\ U_i &= \lambda T_i + (1 - \lambda) C_i, \end{aligned} \tag{13}$$

where $(\beta_{\text{ctx}}, \beta_{\text{hist}}, \beta_{\text{fail}}) = (0.50, 0.30, 0.20)$ and $\lambda = 0.15$. After compatibility filtering, c_i and F_i are constant across eligible records. Their relative order therefore depends on task semantic similarity T_i and recent-history matching H_i. Gripper state is used only for compatibility filtering and does not contribute an additional ranking term.

A.2 Semantic and Visual Encoding

The task semantic encoder is the frozen `Qwen/Qwen3-Embedding-0.6B`. Current and historical task instructions use the same encoding procedure. We take the hidden state of the last valid token, retain its first 256 dimensions, and apply L_2 normalization. Inputs are limited to 256 tokens. Task semantic similarity is

$$T_i = e(\ell)^\top e(\ell_i), \qquad \|e(\ell)\|_2 = \|e(\ell_i)\|_2 = 1. \tag{14}$$

Embeddings for the fixed LIBERO-Pro task instructions are precomputed and cached without using evaluation trajectories or outcomes. An instruction not present in the cache must be encoded and added before retrieval; no alternative text representation is silently substituted.

The visual encoder is the frozen `google/siglip2-base-patch16-224`. Visual ranking compares the current agent-view image with the historical before image using the inner product of unit-normalized embeddings. Historical after images are displayed as evidence in the Memory Card but do not affect ranking. The visual route does not compare sequences of the three most recent images; recent-history matching applies only to primitive sequences.

A.3 Rank Fusion and Evidence Selection

Both routes rank records within the compatible candidate set. The candidate pool is the union of the top $N = 12$ records from each route. Ranks are computed before truncation, with ties broken by record identifier. Rank transformation and fusion are defined as

$$\begin{aligned} \rho(r) &= \frac{c + 1}{c + r}, \\ S_i &= (1 - \alpha)\rho(r_i^{\text{text}}) + \alpha\rho(r_i^{\text{vis}}), \end{aligned} \tag{15}$$

where $r \in \{1, 2, \ldots\}$, $c = 10$, and $\alpha = 0.5$.

Eligible records satisfy

$$S_i \geq \tau_t, \qquad \tau_t = \max(\tau_{\text{abs}}, S_{\max} - \Delta), \tag{16}$$

where $\tau_{\text{abs}} = 0.55$ and $\Delta = 0.15$. This threshold applies to fused rank scores, not task semantic similarity or action success probability. Task semantic similarity contributes to ranking without a separate acceptance threshold.

Records are scanned in descending fused-score order. A record is selected only if neither its next-primitive type nor its source episode has already been selected, up to $K_{\max} = 3$ records. These constraints limit repeated action suggestions and repeated evidence from nearby steps in the same trajectory, respectively. The card is not padded when fewer records qualify. No step-level evidence is added when no record qualifies.

A.4 Memory Card and Existing Memories

For each record, the Memory Card presents the historical task, previous primitive and its local outcome, failure type, gripper state, recent primitive history, and the historical next primitive with its execution outcome. Available before and after images are included. The source episode's final outcome is distinguished from the primitive's local outcome: a successful action does not necessarily imply a successful task.

Retrieved records support the current decision rather than forming an action sequence to execute. The planner selects an action and determines its arguments from the current observation, without directly reusing historical coordinates. CARE-VLA retains Harness VLA's original task-specific and global memories. The planner and VLA parameters remain frozen, and the experience bank remains read-only during evaluation.

B Experience Banks and Execution Settings

LIBERO-Pro. The bank contains 1,093 step-level records from 80 reference episodes across eight suites (Table 5). The frozen VLA backend is the RLinf-released `pi05_libero130_fullshot` $\pi_{0.5}$-SFT checkpoint used by Harness VLA. The 1-suite banks contain the target suite's records; the 4-suite bank pools 532 records from Spat-T/S and Goal-T/S; the 8-suite bank contains all 1,093 records.

RoboTwin C2R. Reference executions are collected in `demo_clean` using one non-evaluation initialization seed per task across all 50 tasks. Evaluation uses five randomized seeds per task (Table 2). Reference executions are segmented at primitive boundaries and stored in a shared bank, allowing retrieval across tasks and episodes. The frozen VLA backend, primitive implementations, execution feedback rules, and remaining execution settings follow Harness VLA (Zhang et al., 2026b).

Shared protocol. Both simulation banks are constructed before evaluation and remain read-only. Evaluation trajectories are never added. Primitive outcomes and failure types use Harness VLA's existing feedback definitions, and task success uses the same benchmark completion criteria. CARE-VLA changes the experience retrieval and evidence supplied to the planner while retaining the original execution interface and memories. Training-free refers to this added retrieval framework; it does not remove the training used to obtain the underlying VLA policies.

C Efficiency Details

Primitive usage. Table 6 reports average primitive executions over all successful episodes and over the 548 task–seed pairs solved by both methods. The shared-success comparison uses the same episode set, whereas each method's all-success average uses its own successful episodes. Failed and timed-out episodes are excluded.

Retrieval timing. We audit the retrieval-core timing recorded in historical execution logs. The timer starts on entry to the retrieval function and stops after evidence selection. It includes task-and-

Table 5: **LIBERO-Pro experience bank statistics.** Each record represents a step-level experience.

Suite	Reference episodes	Step-level records
Spat-T	10	152
Spat-S	10	137
Obj-T	10	74
Obj-S	10	99
Goal-T	10	108
Goal-S	10	135
Long-T	10	191
Long-S	10	197
Total	80	1,093

Table 6: **Primitive usage in successful episodes.** Average numbers of primitive executions across the eight LIBERO-Pro suites. Shared successes are the 548 task–seed pairs solved by both methods. Failed and timed-out episodes are excluded. Lower values are better.

Method	All successes ↓	Shared successes (548) ↓
Harness VLA	15.54	15.34
CARE-VLA	14.06	13.97
Relative reduction	9.5%	8.9%

context scoring, compatibility filtering, the visual encoder service request, visual similarity computation, ranking, fusion, selection, and auxiliary retrieval diagnostics executed within this function. Because the visual encoder request occurs inside the timed interval, its communication and waiting time are included. The reported latency measures experience retrieval and selection, excluding Memory Card rendering, planner inference, and primitive execution.

Across 10,553 recorded retrieval decisions, the decision-weighted average is 93.2 ms. Missing timing values are not counted as zero. The logs span two historical bank snapshots: Goal-T and Spat-S use the earlier snapshot, and the remaining six suites use the later snapshot. These measurements document retrieval-core cost in those runs, rather than a new timing benchmark of the final Qwen-based configuration on the eight-RTX-4090 setup. They do not measure end-to-end task duration.

D Additional Real-Robot Settings

Experiments use a Franka Research 3 robotic arm on three pick-and-place tasks involving an apple, a banana, and a milk container. Observations consist of third-person RGB and depth images, without wrist-camera inputs. For each task, 30 demonstration trajectories are collected to fine-tune a separate task-specific $\pi_{0.5}$ policy. Direct policy execution, Harness VLA, and CARE-VLA use the same corresponding policy. Both agentic methods use GPT-5.5 with xhigh reasoning.

Visual observations. Third-person RGB-D observations are captured by an Orbbec Femto Bolt camera and saved at 5 Hz. The original RGB and depth resolutions are 3840×2160 and 1024×1024, respectively. RGB and depth frames are paired using device timestamps with a maximum allowed difference of 10 ms. Camera intrinsics, distortion parameters, and depth-to-RGB extrinsics are saved with the observations.

Evaluation protocol. Each method is evaluated over 20 trials per task. Object starting positions vary across trials to assess performance under position changes. For both agentic methods, each trial permits at most $T = 40$ primitive executions. A trial succeeds if the object is placed in the designated region within this budget. A trial fails if this limit is reached before task completion, or if a safety stop is triggered. Here, T is a primitive-execution budget, not a wall-clock time limit.

Reference experience. CARE-VLA uses five reference episodes per task, giving 15 episodes that are split into step-level records. For each task, Harness VLA uses one of these five episodes to form

Table 7: **Historical retrieval-core latency.** Values are average milliseconds per recorded retrieval decision. The overall average is weighted by the number of decisions.

Suite	Average latency (ms)
Spat-T	83.4
Spat-S	115.9
Obj-T	87.2
Obj-S	114.9
Goal-T	118.6
Goal-S	83.5
Long-T	77.3
Long-S	77.8
Overall	93.2

its task-specific memory, including an execution trace and a task summary. Both methods retain Harness VLA's global memory, which stores task-independent operating knowledge for the primitive library, including success rules and failure models that guide primitive use and diagnosis before retrying. CARE-VLA adds retrieved step-level evidence while retaining the same task-specific and global memories.