ECHO: Continuous Hierarchical Memory for Vision-Language-Action Models

Yanbin Hu[1,3], **Jin Cui**[2,3], **Jiayi Lu**[3], **Ruixuan Yang**[3], **Jun Ye**[3], **Boran Zhao**[1,3]*, **Xingyu Chen**[2,3], **Xuguang Lan**[2,3], **Pengju Ren**[2,3]

[1]School of Software, Xi'an Jiaotong University
[2]School of Artificial Intelligence, Xi'an Jiaotong University
[3]State Key Laboratory of Human-Machine Hybrid Augmented Intelligence, Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University
yanbinhu@stu.xjtu.edu.cn {boranzhao, pengjuren}@xjtu.edu.cn

Abstract

Memory capacity is a critical factor determining the performance of Vision-Language-Action (VLA) models in long-horizon manipulation tasks. Existing memory-augmented architectures primarily rely on linear or flat storage, lacking structural priors and hierarchical organization of manipulation experience, which hinders efficient retrieval and limits generalization to unseen long-horizon task compositions. Inspired by the hierarchical organization of human experience, we propose ***ECHO*** *(**E**xperience **C**onsolidation and **H**ierarchical **O**rganization)*, a long-term memory framework operating within a Continuous Hierarchical Space. *ECHO* maps VLA hidden states into this space with a hyperbolic autoencoder and organizes experiences into a semantic memory tree using hyperbolic metrics and entailment constraints, enabling efficient top-down retrieval. Background consolidation refines the hierarchy through structural splitting and geometric interpolation, supporting temporary virtual-memory synthesis. Across π_0, $\pi_{0.5}$, and OpenVLA-OFT, *ECHO* consistently improves performance, with up to a 5.42-point gain in average success across the four LIBERO suites. Without any target-suite trajectories in memory, *ECHO* still improves long-horizon execution by transferring and recombining experience from other manipulation suites. This compositional reuse also transfers to five held-out real-robot compositional tasks without complete composite-task trajectories in memory, improving success from 53% to 65%. These results show that continuous hierarchical memory enables transferable experience reuse across policy architectures and improves long-horizon generalization.

1 Introduction

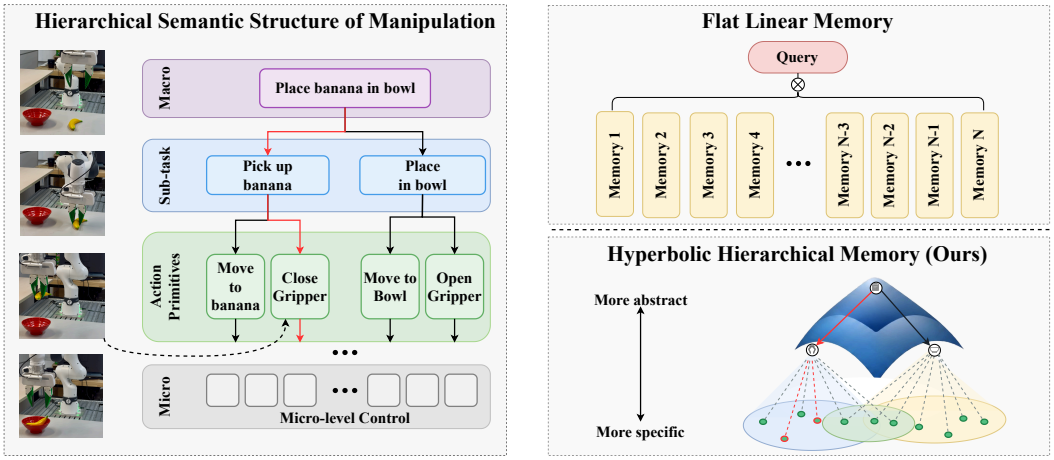
Driven by rapid advancements in cross-modal environmental understanding, Vision-Language-Action (VLA) models such as RT-2 [Zitkovich et al., 2023], OpenVLA [Kim et al., 2024], and π_0 [Black et al., 2024] can directly translate visual observations and text instructions into executable robot actions. During this process, integrating a robust memory system into these foundation models is crucial for achieving advanced embodied intelligence [Shi et al., 2025, Torne et al., 2026, Li et al., 2025a]. Particularly in complex, long-horizon manipulation tasks, the model's ability to extract and utilize past successful experiences often determines the upper bound of decision-making [Wu et al., 2025, Liu et al., 2026, Li et al., 2024]. A key property of such experience reuse is hierarchical

*Corresponding author: boranzhao@xjtu.edu.cn.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

generalization: when humans encounter a new task, they can often transfer memories from similar categories by abstracting common subgoals and action patterns, rather than replaying a previously seen trajectory. This suggests that useful robotic memory should not only store past episodes, but also organize them into reusable abstractions that support generalization across related manipulation categories. Many existing memory-augmented VLA approaches primarily organize experience as flat or temporally indexed memory entries and retrieve relevant history through similarity matching [Shi et al., 2025, Torne et al., 2026, Li et al., 2025a, Ma et al., 2024]. While effective for reusing historical information, such designs typically lack an explicit cross-episode abstraction hierarchy and structured consolidation of related experiences. Consequently, they have difficulty capturing containment relations from abstract skills to concrete instances and extracting reusable common knowledge from dispersed experiences.

Unlike simple linear sequence storage, human experience is organized through hierarchical semantic clustering [Guo et al., 2026, Chen et al., 2025, Kumaran et al., 2016, McClelland et al., 1995]. Robot manipulation actions naturally possess a tree-like containment relationship ranging from macroscopic semantics to microscopic control, as illustrated in Figure 1. This hierarchical nature motivates us to move beyond conventional flat memory organization and design a long-term memory mechanism that explicitly models the relationships among tasks, sub-goals, and actions.



(a) Hierarchical semantics of manipulation tasks

(b) Flat memory vs. hyperbolic hierarchy

Figure 1: **Conceptual overview of the *ECHO* framework.** (a) Manipulation demonstrations exhibit hierarchical semantics, where high-level task instructions are grounded through sub-tasks, action primitives, and micro-level controls. (b) Instead of organizing experiences as a flat linear memory, *ECHO* embeds them into a continuous hyperbolic hierarchical space, enabling coarse-to-fine memory organization, entailment-aware retrieval, and the generation of unseen memories through spatial interpolation.

However, introducing hierarchical memory into VLA models remains challenging. Discrete tree structures easily suffer from combinatorial explosion under fine-grained action variations [Ahn et al., 2022], while continuous spaces require an appropriate geometry to measure distances across different abstraction levels. To address this, we propose a Continuous Hierarchical Space and introduce hyperbolic metrics to organize high-level task semantics and low-level action experiences. Moreover, task relationships in continuous space are not rigid parent-child links, but often partial or overlapping containment relations; for example, placing a banana into a bowl involves both picking and placing. We therefore introduce Hyperbolic Entailment Cones to describe such complex relationships. Finally, the continuous space enables temporary memory priors to be synthesized through geometric interpolation when no precise experience matches the current task. These synthesized representations are used only during inference; only newly observed real experience that leads to successful task completion and passes subsequent validation can be committed to long-term memory. Beyond a single VLA architecture, *ECHO* is designed to augment different base policies, with its effectiveness validated across π_0, $\pi_{0.5}$, and OpenVLA-OFT. The main contributions of this paper are summarized as follows:

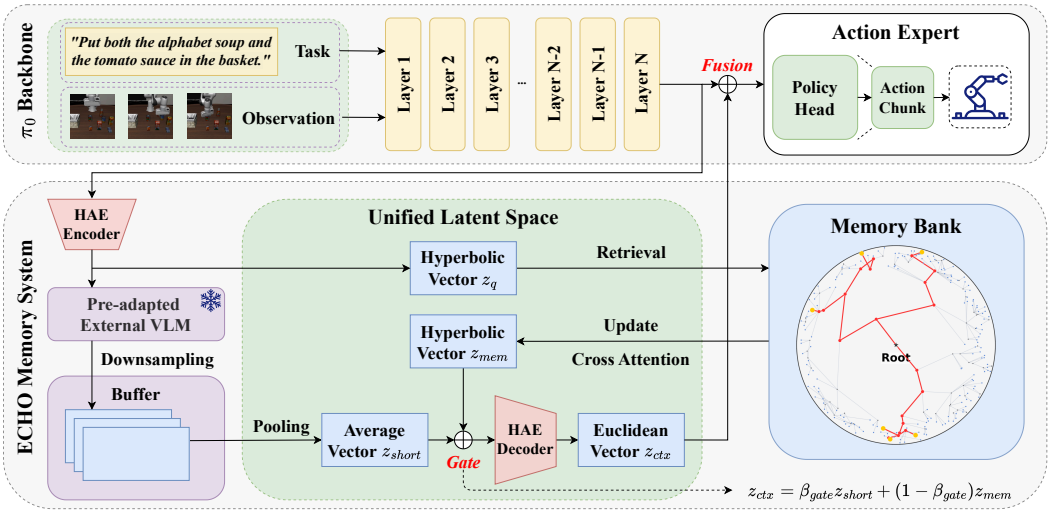


Figure 2: **Overall architecture of *ECHO***, illustrated with π_0 as a representative VLA backbone. Visual-language observations are mapped into a continuous hyperbolic memory space for hierarchical organization and retrieval. Retrieved long-term experience is fused with the current short-term context through a gating mechanism and then used to augment action generation. The same memory framework can be applied to different VLA backbones. The *Unified Latent Space* denotes the Lorentz hyperbolic space produced by the HAE; explicit manifold mappings are omitted for clarity.

- **Continuous hierarchical memory for VLA models.** We propose *ECHO*, a long-term memory framework that organizes manipulation experience in a Continuous Hierarchical Space. Unlike rigid discrete trees, *ECHO* represents task semantics, sub-goals, and action experience in a shared continuous hierarchy and can be adapted across different VLA backbones.
- **Hyperbolic organization with entailment-aware retrieval.** We instantiate the continuous hierarchy with hyperbolic geometry and introduce entailment-cone constraints to model abstraction levels and partial containment relations among manipulation experiences. This enables coarse-to-fine top-down retrieval, avoiding global matching over a flat memory bank.
- **Compositional memory reuse across backbones and tasks.** The continuous memory space allows *ECHO* to reuse and interpolate between related experiences, transferring reusable sub-goal and action priors to new long-horizon task compositions. Experiments across π_0, $\pi_{0.5}$, and OpenVLA-OFT, together with LIBERO cross-suite and target-task-held-out transfer, RMBench evaluation, and real-robot compositional tasks, demonstrate consistent benefits across different base policies and task settings.

2 Related Work

Vision-Language-Action Models. Vision-Language-Action (VLA) models extend Vision-Language Models (VLMs) with action generation capabilities. Existing policies map multimodal inputs to executable actions through diffusion-based decoders [Black et al., 2024, Chi et al., 2025, Team et al., 2024], auto-regressive action-token prediction [Zitkovich et al., 2023, Driess et al., 2023, Kim et al., 2024], or continuous action heads in later adaptations [Kim et al., 2025]. Recent dual-system VLAs, such as DP-VLA, RoboDual, and FiS-VLA [Han et al., 2024, Bu et al., 2024, Chen et al., 2026a], further separate low-frequency semantic reasoning from high-frequency action control to improve reasoning–execution coordination and efficiency. In contrast, *ECHO* does not redesign this reasoning–control decomposition, but introduces a persistent cross-episode memory layer for structured experience reuse across different VLA architectures.

Memory Systems in Embodied Agents. Memory mechanisms have been introduced into embodied agents to improve history awareness and long-horizon decision-making. Early approaches maintain short-term history within the Transformer context window [Brohan et al., 2022, Chen et al.,

2021] or retrieve relevant past experience from external databases [Zhu et al., 2024, Xie et al., 2024, Yoneda et al., 2024], while recent VLA-specific methods introduce stage-wise demonstration memories, perceptual-cognitive memory banks, or multi-scale video/text histories [Li et al., 2025a, Shi et al., 2025, Torne et al., 2026]. These approaches effectively provide temporal and episodic context, but do not explicitly organize persistent cross-episode experience into a general-to-specific hierarchy with entailment-constrained top-down retrieval. *ECHO* instead structurally consolidates manipulation experience into a continuous hierarchy, supporting reusable abstractions across tasks rather than relying only on flat or temporally indexed retrieval.

Hyperbolic Embeddings for Hierarchical Representation. Hyperbolic geometry provides a natural inductive bias for hierarchical representation because negatively curved spaces can embed tree-like structures with low distortion [Sala et al., 2018]. This property has been widely used to model lexical taxonomies such as WordNet [Miller, 1995, Nickel and Kiela, 2017], graph-structured data [Chami et al., 2019, Zhang et al., 2021], and continuous hierarchical latent representations [Mathieu et al., 2019]. *ECHO* brings this geometric prior to VLA memory organization. By mapping task-conditioned hidden states into a hyperbolic space, *ECHO* represents task-level semantics, sub-goals, and action segments in a unified continuous hierarchy, enabling entailment-aware retrieval and structured experience reuse for robotic manipulation.

3 Methodology

ECHO is a hierarchical long-term memory framework for VLA policies. It decomposes successful trajectories into fine-grained sub-goal segments, encodes them into a hyperbolic memory space, and organizes them into a general-to-specific hierarchy. At inference time, *ECHO* retrieves task-relevant experience, optionally composes related memories, and injects the decoded prior into the action-generation pathway through a similarity-modulated residual connection. The memory geometry, retrieval, and consolidation mechanisms are shared across backbones, while only a lightweight projection and injection interface is adapted to the target policy. The overall architecture is shown in Figure 2.

3.1 Experience Abstraction and Memory Construction

To extract reusable experience from continuous control streams, we employ a lightweight VLM-guided semantic segmentation module. The VLM decomposes the task instruction into ordered sub-goals, detects sub-goal transitions during execution, and applies a task-level consistency check after completion to filter failed or invalid trajectories.

Each validated segment forms a fine-grained memory unit containing the local latent state, global task instruction, sub-goal semantics, and the corresponding action chunk or trajectory segment. After hyperbolic encoding, a memory entry is represented as $M_i = (z_i, g_i, s_i, a_i)$, where $z_i \in \mathbb{L}^n$ denotes the hyperbolic embedding and (g_i, s_i, a_i) store the associated task, sub-goal, and action information.

3.2 Hyperbolic Autoencoder and Entailment Constraints

We use a Hyperbolic Autoencoder (HAE) to map Euclidean backbone states into the Lorentz hyperbolic space. The Lorentz manifold with curvature $-c$ ($c > 0$) is defined as $\mathcal{L}^n = \{x \in \mathbb{R}^{n+1} : \langle x, x\rangle_{\mathcal{L}} = -1/c, x_0 > 0\}$ with

$$\langle x, y\rangle_{\mathcal{L}} = -x_0 y_0 + \sum_{i=1}^{n} x_i y_i. \tag{1}$$

The encoder is written compactly as

$$E_\phi(h) = \Pi_{\mathcal{L}}(\mathrm{Exp}_0(f_\phi(h))), \tag{2}$$

where f_ϕ is the Euclidean encoder and $\Pi_{\mathcal{L}}$ projects the representation onto the Lorentz manifold. The decoder D_ψ maps a hyperbolic representation back to the Euclidean backbone feature space.

Hierarchy Definition and Entailment Constraints. Global task embeddings are treated as parent representations z_p, while sub-goal transition states form child representations z_c. The HAE is trained with reconstruction, Lorentzian distance, and entailment constraints:

$$\mathcal{L}_{AE} = \mathcal{L}_{recon}(h, \hat{h}) + \lambda_d d_{\mathcal{L}}(z_p, z_c) + \lambda_e \mathcal{L}_{entail}, \tag{3}$$

where

$$\mathcal{L}_{entail} = \frac{1}{|\mathcal{B}|} \sum_{(z_p, z_c) \in \mathcal{B}} \max(0, \phi(z_c, z_p) - \omega(z_p)) \,. \tag{4}$$

Here, $\phi(z, z_\mu)$ denotes the angular deviation of the tested point z from the entailment cone rooted at z_μ, while $\omega(z_\mu)$ denotes the corresponding cone aperture. The second argument of $\phi(\cdot, \cdot)$ therefore specifies the cone apex and the first denotes the tested point. Task-to-subgoal supervision establishes the initial abstract-to-specific geometric ordering, while deeper hierarchy is subsequently refined through memory consolidation and node splitting.

Compact Representation and Semantic Recovery. After training, experience is stored as compact Lorentz vectors and recovered through D_ψ when required by the policy. This representation provides a structured retrieval space while reducing the storage cost of latent experience compared with retaining the original high-dimensional backbone states.

3.3 Autonomous Memory Consolidation

New experience z_{new} is inserted top-down into the memory hierarchy. A new branch is created when it lies outside the entailment cones of all candidate paths,

$$\forall \mu \in \mathcal{T}_{\mathbb{L}}, \quad \phi(z_{new}, z_\mu) > \omega(z_\mu). \tag{5}$$

Conversely, an existing node is split when a sufficiently large fraction of its children violates the node cone:

$$r_{\text{viol}}(\mu) = \frac{1}{|Z_\mu|} \sum_{z \in Z_\mu} \mathbb{I}[\phi(z, z_\mu) > \omega(z_\mu)] > \tau_{\text{split}}. \tag{6}$$

A Lorentzian K-Means split [Law et al., 2019] is then applied to refine the semantic granularity. Together with the task-to-subgoal entailment supervision, this process progressively forms a deeper general-to-specific memory hierarchy. The resulting global memory structure and depth-wise retrieval patterns are visualized in Appendix B.3, showing the multi-level hierarchy formed after consolidation and its coarse-to-fine traversal during retrieval.

ECHO performs structural consolidation asynchronously during robot idle periods. Newly observed sub-goal segments are first stored in an unconsolidated pool and are committed to the long-term tree only after the corresponding trajectory passes the task-level consistency check, avoiding expensive structural updates during real-time control.

3.4 Hierarchical Retrieval and Geometric Composition

Given the current decision state h_t, the same HAE encoder produces a hyperbolic query $q = E_\phi(h_t)$. For the π_0 instantiation, h_t is the hidden state of the last VLM prefix token; q is used only for retrieval and is not directly injected into the Action Expert.

Hierarchical Retrieval. The system performs top-down beam search over the long-term memory tree and expands a node z_μ only when the query satisfies its entailment cone,

$$\phi(q, z_\mu) \leq \omega(z_\mu). \tag{7}$$

The relevance of a retrieved leaf z is measured by the Lorentzian distance $d_{\mathcal{L}}(z, q) = \operatorname{acosh}(-c\langle z, q \rangle_{\mathcal{L}})/\sqrt{c}$ and converted to $\operatorname{sim}(z, q) = 1/(1 + d_{\mathcal{L}}(z, q))$. The top-$k$ memories are aggregated with softmax-normalized similarity weights to form z_{mem}, while $s_t^\star$ denotes the maximum similarity among the retrieved hits and is later used to modulate memory injection. If no valid memory is retrieved, long-term memory injection is skipped and only the short-term context is retained.

Geometric Memory Synthesis. When no single memory sufficiently matches the current query, related memories can be composed along the Lorentzian geodesic. For two retrieved memories z_i and z_j,

$$z_{syn} = \mathrm{Exp}_{z_i}\big(\rho\,\mathrm{Log}_{z_i}(z_j)\big)\,, \tag{8}$$

where ρ is determined by their normalized query similarities. The synthesized representation is used only as a temporary inference prior and is not permanently inserted into the memory tree; only validated real execution experience is committed afterward.

3.5 Memory-Conditioned Policy Integration

Short-term states are encoded by the HAE and pooled into a local context z_{short}, which is fused with the retrieved long-term prior z_{mem} in the same latent space. We use a scalar similarity gate $\beta_{\mathrm{gate},t} = \sigma(\gamma(\tau - s_t^\star))$ with $\tau = 0.5$ and $\gamma = 5$; the gate is set to one when no valid memory is retrieved, falling back to the short-term context without memory injection. For notational simplicity, intermediate manifold mappings and projections during latent-space fusion are omitted. The fused representation is

$$z_{\mathrm{ctx}} = \beta_{\mathrm{gate},t} z_{\mathrm{short}} + (1 - \beta_{\mathrm{gate},t}) z_{\mathrm{mem}}, \qquad \hat{h}_t = D_\psi(z_{\mathrm{ctx}}). \tag{9}$$

The decoded prior is injected as a residual correction $\Delta_t = \hat{h}_t - h_t$, with dynamic strength $\alpha_t = \alpha_0(1 - \beta_{\mathrm{gate},t})$ (or $\alpha_t = \alpha_0$ when dynamic modulation is disabled). Memory injection is skipped when no valid memory is retrieved. For π_0, the residual is projected to the Action Expert width and broadcast to the suffix tokens:

$$E'_{\mathrm{suffix}} = E_{\mathrm{suffix}} + \mathrm{proj}(\alpha_t \Delta_t)\,. \tag{10}$$

The similarity gate dynamically balances the short- and long-term representations and modulates the subsequent residual injection strength. The same memory construction, geometry, retrieval, and similarity-based modulation are retained across VLA backbones; only the representation projection and injection interface are adapted to the target policy.

4 Experiments

Our empirical evaluation addresses four questions: (1) Does *ECHO* consistently improve different VLA backbones on standard robotic manipulation benchmarks? (2) Can *ECHO* reuse prior experience when target-suite, target-task, or complete composite trajectories are absent from memory? (3) Which components of the hierarchical memory contribute to the final performance? (4) Does the proposed retrieval mechanism remain efficient and scalable as the memory bank grows?

4.1 Experimental Setup

Benchmarks. We use *LIBERO* [Liu et al., 2023] as the primary simulation benchmark, covering Spatial, Object, Goal, and the 10-task Long suite. We further evaluate robustness on *LIBERO-Plus* [Fei et al., 2025], cross-benchmark transfer on the complete nine-task RMBench[Chen et al., 2026b] , and physical manipulation on a Franka Emika Panda robot.

Evaluation protocol. For controlled LIBERO comparisons, the corresponding Base/+ECHO settings use the same backbone checkpoint, evaluation tasks, experience source, and memory budget, and we report mean success rates and standard deviations when repeated evaluations are available. Full evaluation counts, observation inputs, action-chunking, horizons, and stabilization settings are provided in the appendix.

Baselines and memory. Published VLA and memory-augmented methods are included as cross-paper references, while π_0[Black et al., 2024], $\pi_{0.5}$[Physical Intelligence et al., 2025], and OpenVLA-OFT[Kim et al., 2025] are evaluated as matched Base/+*ECHO* pairs. For LIBERO, memory is initialized from 10 official successful demonstrations per task (about 2,000–2,769 verified segments per suite). A fine-tuned Qwen2-VL-7B-Instruct [Wang et al., 2024] is used only for transition detection and trajectory verification, with approximately 90% accuracy on a trajectory-disjoint validation split. For held-out suite/task and real-world compositional evaluations, the memory bank remains frozen throughout evaluation.

Table 1: Success rates (%) on LIBERO, LIBERO-Plus, and RMBench. LIBERO **Avg.** averages the four standard suites, while RMBench reports the nine-task average. Shaded rows denote matched *+ECHO* comparisons. ^{ZS} denotes zero-shot evaluation on LIBERO-Plus.

Method	LIBERO				Avg.	LIBERO-Plus	RMBench Avg.
	Spatial	Object	Goal	Long			
Reported baselines							
Octo [2024]	$78.9_{\pm 1.0}$	$85.7_{\pm 0.9}$	$84.6_{\pm 0.9}$	$51.1_{\pm 1.3}$	75.08	–	–
OpenVLA [2024]	$84.7_{\pm 0.9}$	$88.4_{\pm 0.8}$	$79.2_{\pm 1.0}$	$53.7_{\pm 1.3}$	76.50	15.6	–
CronusVLA [2025b]	93.8	92.8	95.6	86.5	92.18	–	–
MAP-VLA [2025a]	96.3	98.4	95.4	$83.4_{\pm 0.7}$	93.38	–	–
MemoryVLA [2025]	98.4	98.4	96.4	93.4	96.7	70.2	–
ECHO across VLA backbones							
π_0 [2024]	$97.5_{\pm 1.7}$	$97.0_{\pm 1.2}$	$92.3_{\pm 2.5}$	$80.7_{\pm 2.0}$	91.88	54.2^{ZS}	–
π_0 + ***ECHO***	$\mathbf{98.3}_{\pm 1.0}$	$\mathbf{98.8}_{\pm 0.5}$	$\mathbf{98.6}_{\pm 1.0}$	$\mathbf{93.5}_{\pm 2.6}$	**97.30**	$\mathbf{56.9}^{\text{ZS}}$	–
$\pi_{0.5}$ [2025]	99.5	95.5	93.6	91.6	95.05	85.7	22.9
$\pi_{0.5}$ + ***ECHO***	**99.5**	**97.5**	**98.5**	**94.6**	**97.53**	**92.7**	**26.9**
OpenVLA-OFT [2025]	97.6	98.4	97.9	93.3	96.80	69.6	–
OpenVLA-OFT + ***ECHO***	**98.0**	**99.2**	**99.6**	**96.7**	**98.38**	**71.9**	–

4.2 Main Results and Backbone Generality

Table 1 summarizes results on the four standard LIBERO suites and LIBERO-Plus. Published methods provide cross-paper reference points, while the matched Base/*+ECHO* pairs isolate the effect of the memory mechanism. Across π_0, $\pi_{0.5}$, and OpenVLA-OFT, *ECHO* improves the four-suite average by **5.42**, **2.48**, and **1.58** percentage points, respectively.

On π_0, the largest gain occurs on *LIBERO-Long*, from 80.7% to 93.5% (**+12.8** points), highlighting the value of structured experience reuse in long-horizon manipulation. On LIBERO-Plus, π_0*+ECHO* improves the matched backbone from 54.2 to 56.9. Officially reported LIBERO-Plus baseline results are included in Table 1 where available.

4.3 Experience Transfer and Compositional Generalization

Cross-benchmark evaluation. Beyond LIBERO, we evaluate *ECHO* on all nine RMBench tasks using the same adapted $\pi_{0.5}$ checkpoint, training data, random seeds, and evaluation episodes for Base and *ECHO*. Each task is evaluated over 50 episodes under three random seeds. *ECHO* improves $M(1)$ from 30.4% to 35.6%, $M(n)$ from 13.5% to 16.0%, and the overall average from 22.9% to 26.9%.

Held-out experience transfer. We further test whether *ECHO* depends on direct target-task trajectories. For LIBERO-Long, the memory contains only Spatial, Object, and Goal experiences. On RMBench, we use four manipulation-homogeneous $M(1)$ tasks—*Rearrange Blocks*, *Put Back Block*, *Swap Blocks*, and *Swap T*—to keep memory complexity fixed while testing cross-task transfer. For each target task, its trajectories are removed and the memory is constructed from the other three tasks with a matched segment count. Under this target-task-held-out setting, *ECHO* improves success from 31.0% to 33.5%, compared with 36.9% when same-target memory is available. For real-world composition, the memory contains atomic-skill demonstrations but no complete composite trajectory, composite instruction, or evaluation layout, providing a further test of whether stored experience can be recomposed for unseen long-horizon tasks.

Suite-level transfer. With 300 official successful demonstrations from LIBERO-Spatial, Object, and Goal and no LIBERO-Long trajectory in memory, *ECHO* improves Long success from 80.70% to 89.31%, compared with 93.48% when same-suite Long memory is available. Thus, a substantial portion of the long-horizon gain remains even when direct target-suite experience is removed.

Retrieval-path analysis. To verify that the improvement comes from compositional reuse rather than direct replay, we visualize a representative successful rollout in Figure 3. The retrieved memories do not correspond to an exact target-task trajectory; instead, *ECHO* activates reusable source-

Table 2: Experience transfer under target-memory holdout. Success rates are reported in percentage points. "Held-out memory" excludes target-suite, target-task, or complete composite-task experience according to the evaluation setting; "same-target memory" provides the corresponding reference when available.

Evaluation setting	Base success	ECHO: held-out memory	ECHO: same-target memory
LIBERO suite-level transfer	80.70	89.31	93.48
RMBench four-task transfer	31.0	33.5	36.9
Real-robot composition	53.0	65.0	–

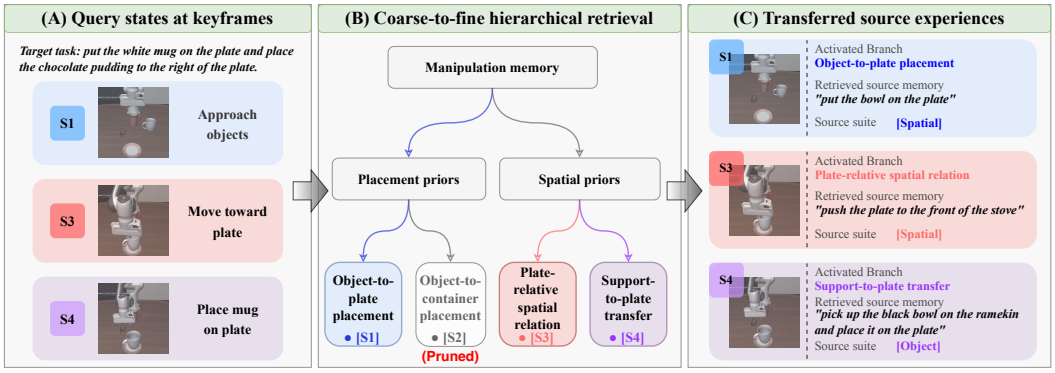


Figure 3: Retrieval-path evidence for cross-suite compositional memory reuse. (A) Representative query states from a successful *ECHO* rollout. (B) Coarse-to-fine traversal of the hierarchical memory tree, showing both activated and pruned branches. (C) Representative memories retrieved from non-target suites along the activated paths, illustrating reusable structural priors rather than exact target-task replay. Branch names are post-hoc semantic summaries for visualization.

memory branches, including object-to-plate placement, plate-relative spatial relation, and support-to-plate transfer priors. This suggests that the hierarchical memory tree supports source-suite experience reuse for unseen long-horizon compositions. The complete top-k retrieval records are provided in Appendix B.4.

Real-world deployment. We further validate *ECHO* on a Franka Emika Panda robot using three representative tabletop manipulation tasks: *Place Banana in Bowl*, *Stack Blocks*, and *Insert Circle into Base*, covering semantic goal grounding, sequential manipulation, and precision-sensitive insertion. Each task is evaluated over 20 independent trials with randomized initial object configurations. *ECHO* consistently improves all three tasks and raises the average success rate from 58.3% to 70.0%, showing that hierarchical memory augmentation remains effective under real visual and control variations.

Table 3: Success rates (%) in real-world manipulation tasks.

Model	Place	Stack	Insert	Avg.
Vanilla π_0	75.0	45.0	55.0	58.3
***ECHO* (Ours)**	**90.0**	**55.0**	**65.0**	**70.0**

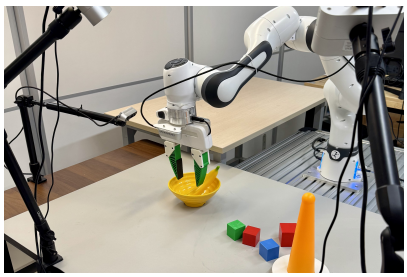


Figure 4: Real-world experimental setup.

Real-world compositional transfer. We further evaluate five held-out compositional tasks (G1–G5) on a Franka Emika Panda robot, covering multi-step placement, object sorting, stacking, and loaded-object transport. The robot-policy checkpoint and training data are unchanged. Memory is constructed from only 10 successful teleoperated demonstrations per atomic subtask under standard

Table 4: Ablation on *LIBERO-Long*. Hyperbolic organization and hierarchical retrieval account for the major improvement, while consolidation and virtual synthesis provide further gains under the same backbone and fusion budget.

Model Variant	Memory / Retrieval Configuration	Success (%)
Vanilla π_0	–	80.70
π_0 + Short-term Buffer	Euclidean within-episode buffer	88.81
π_0 + Flat Memory	Flat Euclidean kNN retrieval	83.25
π_0 + Hyperbolic Memory	Hyperbolic distance-based kNN	91.11
π_0 + Cone Tree	Entailment-aware cone-tree retrieval	92.04
π_0 + *ECHO* w/o Virtual Synthesis	Cone tree + consolidation	92.69
Full *ECHO*	**Cone tree + consolidation + virtual synthesis**	**93.48**

Table 5: Memory-scale analysis on *LIBERO-Long*. Tree and kNN denote retrieval latency for the hierarchical cone tree and flat Euclidean search, respectively. Success rates in this scaling study are from a single evaluation run.

Memory	**Segments**	**Success (%)**	**Tree (ms)**	**kNN (ms)**
25%	692	91.67	4.9	7.8
50%	1,384	92.25	5.5	15.3
75%	2,077	93.25	5.8	22.1
100%	2,769	93.50	6.1	29.5

layouts, with no complete composite trajectory, composite instruction, or evaluation layout stored. Each task uses 20 paired trials, yielding 100 matched initial configurations and 200 physical rollouts across the backbone and *ECHO*. No retry, online policy adaptation, or memory update is allowed during evaluation. Under this protocol, *ECHO* improves overall success from 53% to 65%. Per-task compositional results and full evaluation details are provided in Appendix C.

4.4 Ablation Study

We ablate *ECHO* on *LIBERO-Long* under the same π_0 backbone, memory budget, fusion module, and residual injection interface. The variants compare different memory organizations and retrieval mechanisms, ranging from within-episode buffering and flat Euclidean retrieval to hyperbolic organization, hierarchical cone-tree retrieval, consolidation, and virtual synthesis.

A short-term buffer improves the backbone from 80.70% to 88.81%, whereas flat Euclidean kNN long-term memory reaches only 83.25%, indicating that simply introducing external memory is insufficient. Organizing long-term experience in hyperbolic space substantially raises success to **91.11%**, and entailment-aware cone-tree retrieval further improves it to **92.04%**. Adding memory consolidation further improves success to 92.69%, while virtual synthesis contributes an additional +0.79 percentage points, yielding **93.48%** with the full *ECHO*.

4.5 Efficiency and Scalability

We finally analyze how memory scale affects both task success and retrieval cost. Retaining 25%, 50%, 75%, and 100% of the Long trajectories yields 692–2,769 verified segments. As coverage increases, success improves from 91.67% to 93.50%, with diminishing gains at larger scales. This trend suggests that once key state transitions are covered, additional repeated experience provides smaller marginal benefits than expanding experience diversity.

Cone-tree retrieval scales substantially better than flat kNN in this regime. As the memory bank grows from 692 to 2,769 segments, cone-tree retrieval increases only from 4.9 ms to 6.1 ms, whereas flat kNN grows from 7.8 ms to 29.5 ms. The hierarchy therefore serves not only as a semantic organization mechanism, but also as an efficient access structure for a growing long-term memory bank.

5 Conclusion

We presented *ECHO*, a continuous hierarchical memory framework for VLA models. By organizing successful manipulation experiences in a hyperbolic memory space and retrieving them through entailment-aware tree search, *ECHO* provides reusable long-term priors while preserving the stability of the base π_0 policy through residual injection. Experiments on LIBERO, cross-suite generalization, ablations, and preliminary real-world tasks show consistent gains, especially on long-horizon manipulation. Future work will study larger-scale lifelong memory banks, more robust memory verification, and continual deployment in open-ended environments.

Acknowledgments and Disclosure of Funding

This work was supported in part by Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China under Grant JYB2025XDXM504, and National Natural Science Foundation of China No.62302381, No.52441602. The Authors are with the State Key Laboratory of Human-Machine Hybrid Augmented Intelligence, Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an, Shaanxi, China.

References

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

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. *arXiv preprint arXiv:2508.19236*, 2025.

Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z Ren, Haohuan Wang, Jiaming Tang, Kyle Stachowicz, et al. Mem: Multi-scale embodied memory for vision language action models. *arXiv preprint arXiv:2603.03596*, 2026.

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 preprint arXiv:2511.09516*, 2025a.

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. *arXiv preprint arXiv:2510.10181*, 2025.

Zhen Liu, Xinyu Ning, Zhe Hu, Xinxin Xie, Weize Li, Zhipeng Tang, Chongyu Wang, Zejun Yang, Hanlin Wang, Yitong Liu, et al. Goal2skill: Long-horizon manipulation with adaptive planning and reflection. *arXiv preprint arXiv:2604.13942*, 2026.

Zaijing Li, Yuquan Xie, Rui Shao, Gongwei Chen, Dongmei Jiang, and Liqiang Nie. Optimus-1: Hybrid multimodal memory empowered agents excel in long-horizon tasks. *Advances in neural information processing systems*, 37:49881–49913, 2024.

Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision-language-action models for embodied ai. *arXiv preprint arXiv:2405.14093*, 2024.

Liangxuan Guo, Haoyang Chen, Yang Chen, Yanchao Bi, and Shan Yu. A neural network for modeling human concept formation, understanding and communication. *Nature Computational Science*, pages 1–15, 2026.

Jiaxuan Chen, Yu Qi, Yueming Wang, and Gang Pan. Human-like cognitive generalization for large models via brain-in-the-loop supervision. *arXiv preprint arXiv:2505.09085*, 2025.

Dharshan Kumaran, Demis Hassabis, and James L McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, 20 (7):512–534, 2016.

James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

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

ByungOk Han, Jaehong Kim, and Jinhyeok Jang. A dual process vla: Efficient robotic manipulation leveraging vlm. *arXiv preprint arXiv:2410.15549*, 2024.

Qingwen Bu, Hongyang Li, Li Chen, Jisong Cai, Jia Zeng, Heming Cui, Maoqing Yao, and Yu Qiao. Towards synergistic, generalized, and efficient dual-system for robotic manipulation. *arXiv preprint arXiv:2410.08001*, 2024.

Hao Chen, Jiaming Liu, Chenyang Gu, Zhuoyang Liu, Renrui Zhang, Xiaoqi Li, Xiao He, Yandong Guo, Chi-Wing Fu, Shanghang Zhang, et al. Fast-in-slow: a dual-system vla model unifying fast manipulation within slow reasoning. *Advances in Neural Information Processing Systems*, 38: 98049–98083, 2026a.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Yichen Zhu, Zhicai Ou, Xiaofeng Mou, and Jian Tang. Retrieval-augmented embodied agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17985–17995, 2024.

Quanting Xie, So Yeon Min, Pengliang Ji, Yue Yang, Tianyi Zhang, Kedi Xu, Aarav Bajaj, Ruslan Salakhutdinov, Matthew Johnson-Roberson, and Yonatan Bisk. Embodied-rag: General non-parametric embodied memory for retrieval and generation. *arXiv preprint arXiv:2409.18313*, 2024.

Takuma Yoneda, Jiading Fang, Peng Li, Huanyu Zhang, Tianchong Jiang, Shengjie Lin, Ben Picker, David Yunis, Hongyuan Mei, and Matthew R Walter. Statler: State-maintaining language models for embodied reasoning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15083–15091. IEEE, 2024.

Frederic Sala, Chris De Sa, Albert Gu, and Christopher Ré. Representation tradeoffs for hyperbolic embeddings. In *International conference on machine learning*, pages 4460–4469. PMLR, 2018.

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11): 39–41, 1995.

Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems*, 30, 2017.

Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. *Advances in neural information processing systems*, 32, 2019.

Yiding Zhang, Xiao Wang, Chuan Shi, Xunqiang Jiang, and Yanfang Ye. Hyperbolic graph attention network. *IEEE Transactions on Big Data*, 8(6):1690–1701, 2021.

Emile Mathieu, Charline Le Lan, Chris J Maddison, Ryota Tomioka, and Yee Whye Teh. Continuous hierarchical representations with poincaré variational auto-encoders. *Advances in neural information processing systems*, 32, 2019.

Marc Law, Renjie Liao, Jake Snell, and Richard Zemel. Lorentzian distance learning for hyperbolic representations. In *International conference on machine learning*, pages 3672–3681. PMLR, 2019.

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

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

Tianxing Chen, Yuran Wang, Mingleyang Li, Yan Qin, Hao Shi, Zixuan Li, Yifan Hu, Yingsheng Zhang, Kaixuan Wang, Yue Chen, et al. Rmbench: Memory-dependent robotic manipulation benchmark with insights into policy design. *arXiv preprint arXiv:2603.01229*, 2026b.

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

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

Hao Li, Shuai Yang, Yilun Chen, Yang Tian, Xiaoda Yang, Xinyi Chen, Hanqing Wang, Tai Wang, Feng Zhao, Dahua Lin, et al. Cronusvla: Transferring latent motion across time for multi-frame prediction in manipulation. *arXiv e-prints*, pages arXiv–2506, 2025b.

A Implementation Details

A.1 Hyperbolic Autoencoder Architecture and Training

The Hyperbolic Autoencoder (HAE) is implemented as a lightweight MLP encoder–decoder. The encoder maps a backbone hidden state to a 512-dimensional Euclidean latent feature and then applies the exponential map to obtain a Lorentz representation. The decoder first maps the Lorentz representation back to the tangent space and then reconstructs a Euclidean feature compatible with the backbone-specific injection interface. Accordingly, both short-term context and long-term memory are represented in the same hyperbolic latent space before policy integration.

The HAE is optimized with AdamW using a learning rate of 1×10^{-3}, weight decay of 1×10^{-6}, and batch size 256. The Lorentz representation uses $n = 512$ spatial dimensions plus the time-like coordinate. Consistent with Section 3.2, the training objective combines Euclidean reconstruction, Lorentzian task–subgoal distance regularization, and the entailment penalty; no additional loss term is used in the appendix formulation.

A.2 Memory Tree Construction and Consolidation

The long-term memory tree is refined with Lorentzian K-Means using binary splits ($K = 2$). For a node z_μ with child set Z_μ, we use the same violation criterion as in the main text,

$$r_{\text{viol}}(\mu) = \frac{1}{|Z_\mu|} \sum_{z \in Z_\mu} \mathbb{I}[\phi(z, z_\mu) > \omega(z_\mu)] \,. \tag{11}$$

A node is split when $r_{\text{viol}}(\mu) > \tau_{\text{split}}$ with $\tau_{\text{split}} = 0.3$. This prevents a broad branch from accumulating geometrically incompatible experience and progressively refines the hierarchy from coarse task-level concepts to more specific execution patterns.

Structural consolidation is executed asynchronously outside the real-time control loop. Newly observed sub-goal segments are first stored in an unconsolidated pool and are committed to the long-term tree only after the corresponding trajectory passes task-level consistency verification. Consolidation then performs node splitting and removes redundant entries when needed. During an episode, the policy parameters and long-term tree remain fixed; only validated successful real experience can be considered for later insertion. For the held-out suite/task and real-world compositional evaluations, the memory bank is frozen for the entire evaluation.

A.3 Hierarchical Retrieval and Policy Integration

Given the current decision state h_t, the HAE produces the hyperbolic query $q = E_\phi(h_t)$. For the π_0 instantiation, h_t is the hidden state of the final VLM prefix token. Top-down beam search expands a memory node z_μ only when the query lies inside its entailment cone,

$$\phi(q, z_\mu) \leq \omega(z_\mu). \tag{12}$$

We use beam width $k = 3$. Retrieved leaves are ranked with Lorentzian distance and converted to similarity scores; the top-k entries are aggregated with softmax-normalized similarity weights to obtain the long-term prior z_{mem}. When no single entry provides a sufficiently suitable match, the geometric synthesis module may form a temporary geodesic prior as described in Section 3.4. Such synthesized memories are inference-time candidates only and are never directly committed to the long-term tree.

Short-term states are encoded by the same HAE and pooled into z_{short}. Following Section 3.5, short- and long-term context are fused with a scalar similarity gate

$$\beta_{\text{gate},t} = \sigma\left(\gamma(\tau - s_t^\star)\right), \qquad \tau = 0.5,\ \gamma = 5. \tag{13}$$

When no valid memory is retrieved, the gate is set to one, preserving the short-term context and skipping memory injection. The resulting context is decoded back to the backbone feature space and injected through the backbone-specific residual interface. The dynamic residual strength is $\alpha_t = \alpha_0(1 - \beta_{\text{gate},t})$ with $\alpha_0 = 0.03$ in the main experiments. Thus, the appendix uses the same scalar-gating formulation as the main method rather than a separate learned MLP gate.

A.4 VLM-guided Subgoal Extraction and Verification

We use Qwen2-VL-7B-Instruct as the evaluator for sub-goal decomposition, transition detection, and final trajectory verification. The evaluator is LoRA-tuned on real LIBERO demonstrations. Supervision is constructed from the official HDF5 MuJoCo states and BDDL task predicates: frames satisfying the relevant predicate are treated as positive transition/success examples, while earlier frames from the same successful trajectory provide negative examples. No synthetic rollout, LLM-generated label, or self-labeling is used.

Training and deployment use the same structured prompts and JSON output schema, and the data split is trajectory-disjoint. The resulting evaluator achieves approximately 90% mean accuracy on the held-out validation split. It is used only for memory construction and verification and never generates robot actions.

A.5 Implementation Hyperparameters

Table 6 summarizes the main implementation parameters that are omitted from the main text for brevity.

Table 6: Main implementation hyperparameters of *ECHO*.

Quantity	Value
HAE spatial latent dimension	512
HAE optimizer	AdamW
HAE learning rate	1×10^{-3}
HAE weight decay	1×10^{-6}
HAE batch size	256
Lorentzian split clusters	$K = 2$
Split threshold τ_{split}	0.3
Retrieval beam width	$k = 3$
Similarity-gate threshold τ	0.5
Similarity-gate sharpness γ	5
Base residual strength α_0	0.03

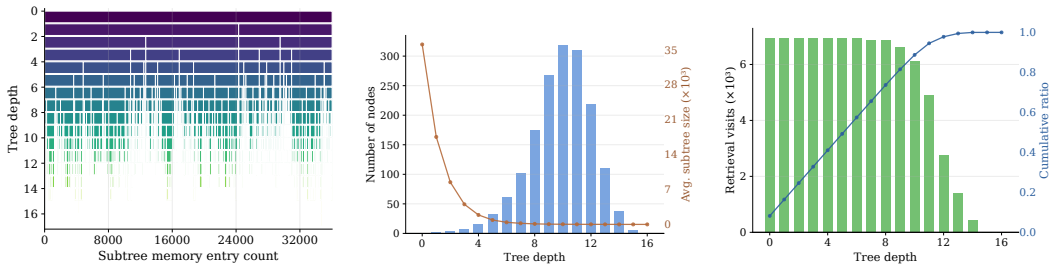
B Experimental Protocols and Additional Analysis

B.1 LIBERO Evaluation Protocol

For LIBERO-Spatial, Object, Goal, and the official `libero_10` (Long) suite, the policy receives 224×224 third-person and wrist RGB observations, the language instruction, and 8-D proprioception. The policy executes five predicted actions per replan. Maximum horizons are 220/280/300/520 steps for Spatial/Object/Goal/Long, respectively, followed by 10 stabilization steps, with no retry after failure.

Each task is evaluated for 50 episodes under seeds 7, 123, and 2024 using identical official initial states and success predicates for the matched Base/+*ECHO* comparisons. Vanilla π_0 and its *ECHO* variant use the same 30k-step checkpoint. Across ablations, the backbone, memory entries, retrieval budget, fusion/injection interface, evaluation episodes, random seeds, and initial states are fixed.

The initial memory for each standard suite is constructed from 10 official successful demonstrations per task, i.e., 100 trajectories per suite, yielding approximately 2,000–2,769 verified sub-goal segments. The policy and memory tree are frozen within each episode; afterward, only successful real segments that pass the consistency check may be used for subsequent memory updates. In the target-held-out evaluations described below, the memory bank is frozen throughout the entire evaluation.



(a) Consolidated memory tree (b) Depth statistics (c) Retrieval visits

Figure 5: Global organization of the consolidated *ECHO* memory bank. (a) Icicle visualization of the depth-truncated cone memory tree, where bands denote tree depths and segment widths indicate subtree memory counts. (b) Node counts and average subtree sizes across depths, illustrating the progressive refinement from broad clusters to finer branches. (c) Retrieval visits across tree depths, showing that *ECHO* traverses multiple hierarchy levels during coarse-to-fine retrieval rather than performing global matching over a flat memory pool.

B.2 Held-out Experience Protocols

The suite-level transfer experiment on LIBERO-Long uses only the 300 official successful demonstrations from LIBERO-Spatial, Object, and Goal; no LIBERO-Long trajectory is present in memory. The task-level RMBench transfer experiment removes all trajectories from the target task and constructs memory only from the remaining tasks with a matched segment count. For the complete nine-task RMBench evaluation, Base and +*ECHO* use the same adapted $\pi_{0.5}$ checkpoint and each task is evaluated over 50 episodes under three random seeds.

These protocols separate direct retrieval of target-task experience from transfer of reusable state-transition priors. They also ensure that the held-out gains reported in Section 4.3 cannot be explained by writing evaluation trajectories back into memory during testing.

B.3 Global Memory Structure Analysis

To examine whether memory consolidation produces the intended hierarchical organization, we analyze the global tree structure and retrieval-access distribution of the consolidated memory bank. As shown in Figure 5, consolidation organizes experience into a multi-level branching structure whose subtrees become progressively finer with depth. Retrieval visits likewise span multiple hierarchy levels, indicating that inference follows coarse-to-fine traversal over the learned tree rather than degenerating into global matching over a flat memory pool.

B.4 Additional Qualitative Retrieval Case

This appendix provides the full top-k retrieval records for the qualitative case study in Figure 3. The target task is *put the white mug on the plate and put the chocolate pudding to the right of the plate*. *ECHO* succeeds in this episode while the corresponding baseline fails. Retrieved memories are task-level source memories from the retrieval trace; branch labels in the main figure are post-hoc summaries for readability.

Table 7: Full top-k retrieval records for the qualitative cross-suite generalization case.

Step	Semantic keyframe	Top-1 retrieved memory	Top-2 retrieved memory	Top-3 retrieved memory	Similarity
S1	Approach objects	put the bowl on the plate ID: 2398	put the bowl on the plate ID: 1956	put the wine bottle on the rack ID: 2697	1.00 / 1.00 / 1.00
S2	Grasp mug	put the bowl on the plate ID: 484	put the bowl on the plate ID: 2443	put the wine bottle on top of the cabinet ID: 2598	0.98 / 0.98 / 0.97
S3	Move toward plate	push the plate to the front of the stove ID: 1299	put the wine bottle on top of the cabinet ID: 997	put the bowl on the stove ID: 211	1.00 / 1.00 / 1.00
S4	Place mug on plate	pick up the black bowl on the ramekin and place it on the plate ID: 220	pick up the black bowl on the ramekin and place it on the plate ID: 572	pick up the black bowl on the ramekin and place it on the plate ID: 941	0.99 / 0.99 / 0.99
S5	Final arrangement	pick up the black bowl on the ramekin and place it on the plate ID: 220	pick up the black bowl on the ramekin and place it on the plate ID: 572	pick up the black bowl on the ramekin and place it on the plate ID: 941	0.99 / 0.99 / 0.99

B.5 Hardware and Software Specifications

Simulation evaluation and offline memory training were conducted on shared Linux workstations equipped with 8 NVIDIA RTX 6000 Ada GPUs, dual-socket AMD EPYC 7H12 CPUs, and 2.0 TiB host memory. The software environment uses PyTorch 2.7.1+cu126, CUDA runtime 12.6, and JAX 0.5.3. Memory-bank construction, including graph preload, takes approximately 4.1–4.6 minutes per run on the same workstation.

Table 8: Simulation hardware and software specifications.

Quantity	**Specification**
Simulation accelerator(s)	8 NVIDIA RTX 6000 Ada Generation GPUs, each with 49,140 MiB VRAM
Host CPU	AMD EPYC 7H12 64-Core Processor; 256 logical cores
Host RAM	2.0 TiB
Software stack	PyTorch 2.7.1+cu126; CUDA runtime 12.6; NVIDIA driver-reported CUDA 13.0
Memory preprocessing	Memory-bank construction including graph preload: approximately 4.1–4.6 minutes per run

C Real-World Experimental Details

This section supplements the real-world results in Section 4.3. The quantitative table and physical setup figure for the three basic tabletop tasks are reported in the main text; here we retain the deployment protocol, the complete 12-frame rollout visualization, and the full held-out compositional evaluation.

C.1 Hardware Platform and Basic-Task Protocol

Real-world evaluations use a Franka Emika Panda robotic arm with a parallel gripper and a static third-person RGB camera. The control loop targets 15 Hz. RGB observations are padded and resized to 224×224 before policy inference, preserving aspect ratio and avoiding cropping artifacts. The policy server runs remotely through a websocket interface, while proprioceptive states and gripper signals are streamed to match the simulation-side policy input format.

The basic physical evaluation uses *Place Banana in Bowl*, *Stack Blocks*, and *Insert Circle into Base*, covering semantic goal grounding, sequential manipulation, and precision-sensitive insertion. Each task is evaluated over 20 independent trials with randomized initial object configurations. The corresponding success-rate table and physical setup figure are moved to the main experimental section, while the complete rollout sequences remain in this appendix.

C.2 Held-out Compositional Manipulation

To test compositional experience reuse beyond the three basic tasks, we construct five held-out manipulation tasks: **G1**, placing a mug on a plate and then inserting a block into the mug; **G2**,

arranging a bowl and a mug on opposite sides of a tray and then placing a small object into the bowl; **G3**, sorting two objects into different containers; **G4**, placing one block on a plate and stacking another block on top of it; and **G5**, placing an object into a bowl and then transporting the loaded bowl to a tray.

Memory construction. For each atomic subtask, we collect 10 successful teleoperated demonstrations under standard layouts and independently segment and validate them before memory construction. Shared atomic skills reuse the same demonstration pool across task groups. The memory bank contains no complete composite-task trajectory, composite instruction, or evaluation layout; thus successful execution requires recombining previously observed atomic experience rather than replaying a stored target-task trajectory.

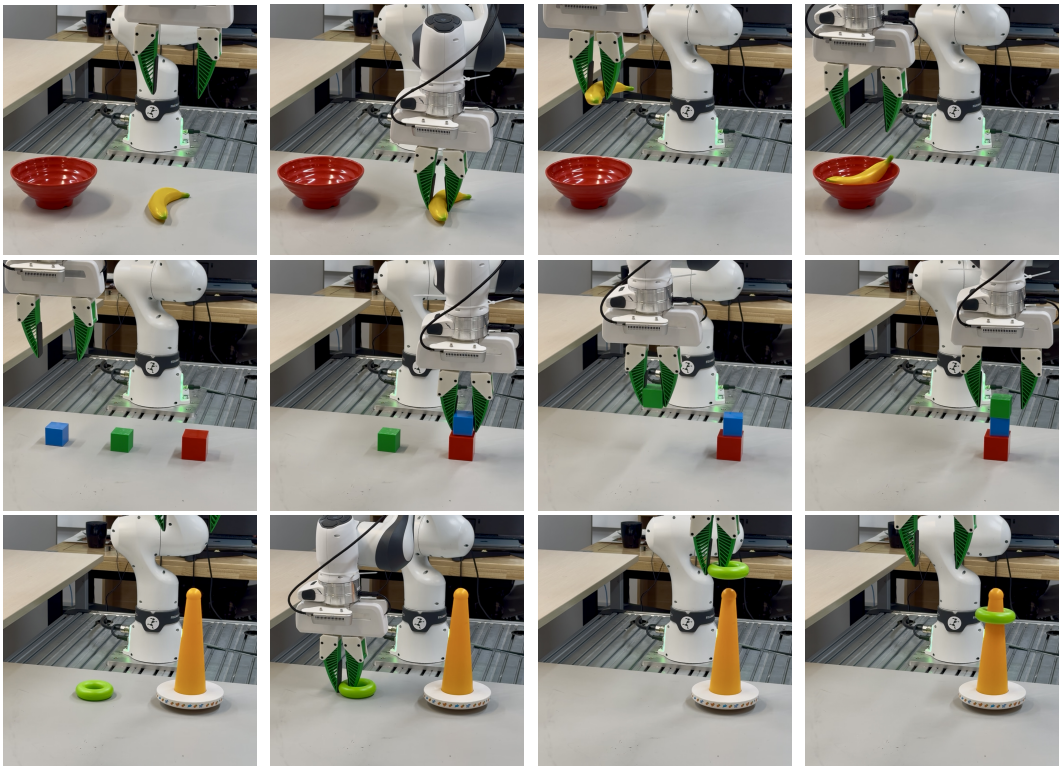


Figure 6: Real-world execution sequences of *ECHO* on three tabletop manipulation tasks: pick-and-place, block stacking, and ring insertion. Each row shows one complete rollout on the Franka platform.

Evaluation protocol. Each compositional task is evaluated with 20 paired trials, yielding 100 matched initial configurations and 200 physical rollouts across the backbone and *ECHO*. For each configuration, both methods are executed once from identical initial object states. Evaluation positions, orientations, distractors, and valid subtask orderings are held out from memory construction. No retry, online policy adaptation, or memory update is allowed, and the long-term memory tree remains frozen throughout the entire evaluation. Success is determined by predefined task-completion predicates and verified from the recorded physical rollouts.

Across the five held-out tasks, *ECHO* improves overall success from 53% to 65%. Because neither complete target compositions nor evaluation layouts are present in memory, this result complements the suite- and task-level held-out studies in the main text with physical evidence of atomic-experience recombination.

Table 9: Held-out real-world compositional evaluation. Each task uses 20 paired trials.

Task	Backbone	*+ECHO*
G1	70% (14/20)	**80% (16/20)**
G2	50% (10/20)	**55% (11/20)**
G3	60% (12/20)	**75% (15/20)**
G4	40% (8/20)	**55% (11/20)**
G5	45% (9/20)	**60% (12/20)**
Overall	**53%**	**65%**

D Limitations

While *ECHO* consistently improves long-horizon and held-out manipulation, its performance still depends on the quality of semantic transition extraction and on whether relevant prior experience exists in the memory bank. The evaluator and task predicates reduce memory pollution, and entailment-constrained retrieval together with similarity-modulated injection limits the effect of poorly matched entries, but these mechanisms do not eliminate retrieval errors completely.

The real-world experiments remain controlled tabletop evaluations. They include randomized layouts, real visual/control noise, and held-out task compositions, but do not cover fast moving objects, severe occlusion, external intervention, or fully open-ended household interaction. In such settings, performance remains bounded by the perception and closed-loop control capabilities of the underlying VLA backbone. Scaling to larger lifelong memory banks and more diverse physical environments remains important future work.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 describe the hierarchical memory method and its evaluated scope. Section 4 supports the claims with matched backbone comparisons, held-out experience transfer, and controlled real-robot evaluations; Appendix D states the limits of generalization.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Appendix D discusses dependence on transition extraction and relevant prior experience, residual retrieval errors, and the restriction of physical evaluations to controlled tabletop settings.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper introduces an empirical memory-augmentation method and does not state theoretical results requiring proofs. The equations in Section 3 define the model and algorithms.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 3 and Appendices A–B provide the algorithmic details, implementation configurations, hyperparameters, and evaluation protocols needed to reproduce the main experimental results. We also plan to publicly release our implementation code, relevant data, and reproduction instructions to further facilitate reproducibility.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The implementation code, trained checkpoints, and newly collected robot data are not publicly available at this stage due to ongoing follow-up research. The paper documents the benchmarks and experimental protocols. Upon completion of the follow-up research, we plan to publicly release the implementation code, relevant data, trained checkpoints, and reproduction instructions to support independent reproduction of the results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4.1 and Appendices A–B specify major optimizer settings, memory parameters, evaluation seeds, and episode counts.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: Section 4.1 explicitly defines the reported variability as standard deviations, and Table 1 provides the corresponding standard deviations for the matched π_0 comparisons to characterize experimental variability.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix B.5 details the GPU and CPU models, device and host memory configurations, software environment, and memory-bank construction time. Additional efficiency experiments evaluate the computational overhead and execution efficiency of the proposed memory framework, providing further information on the computational resources required for reproduction.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and confirm that our research conforms to its requirements. The study focuses on robotic manipulation using established benchmarks and controlled real-robot experiments, following applicable research integrity, safety, and data-use practices.

   Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This work focuses on hierarchical memory organization and experience reuse for vision-language-action models, representing foundational research in robot learning. The experiments are conducted on research benchmarks and in controlled real-robot settings, without deployment in societal applications or identifiable direct societal impacts at this stage.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This work focuses on hierarchical memory augmentation for robotic manipulation and does not currently release new models or datasets posing high risks of misuse. Therefore, the release-specific safeguards addressed by this question are not applicable.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Existing benchmarks and pretrained policy backbones are properly credited in the related-work section and Section 4. LIBERO code and datasets are provided under MIT and CC BY 4.0, respectively, while OpenPI and OpenVLA-OFT code are provided under Apache 2.0 and MIT licenses. The model checkpoints are used in accordance with their applicable terms.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new code, model checkpoints, or datasets are publicly released at this stage, as follow-up research is ongoing. The proposed method and experimental setup are documented in the paper and appendix. We plan to release the relevant assets with appropriate documentation and reproduction instructions upon completion of the follow-up research.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

Justification: This work involves simulated and real-robot manipulation experiments without crowdsourcing or human-subject studies.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This work does not involve crowdsourcing or research with human subjects. All simulated and real-robot experiments evaluate robotic manipulation performance and task completion, and therefore do not require human-subject research approval.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: Section 4.1 and Appendix A.4 document the use of Qwen2-VL-7B-Instruct for subgoal decomposition, transition detection, and trajectory verification, including LoRA fine-tuning with predicate-derived supervision. The VLM supports memory construction and verification without directly generating robot actions.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.