EVA: A MEMORY-AUGMENTED HARNESS FOR LONG-HORIZON EMBODIED TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in Vision-Language-Action (VLA) models have improved the generalization of embodied foundation models, yet their performance remains limited on tasks requiring long-horizon reasoning, persistent memory, and robust execution under environmental noise. Moreover, consolidating diverse capabilities into a single model is costly and difficult to scale. Inspired by general-purpose agent harnesses, recent works have explored agentic frameworks for embodied tasks, but these approaches largely rely on directly adapting general-purpose harnesses, leaving the design of a dedicated, efficient, and general-purpose embodied harness largely unexplored. Therefore, we introduce **EVA**, a general and extensible embodied harness that provides a lightweight set of essential components, integrating perception, memory, memory-grounded planning, execution, monitoring, and verification. At each step, EVA generates an executable subtask and selects an appropriate executor, monitors its execution online, verifies the outcome, and updates memory to enable replanning or recovery. This decoupled design enables existing embodied models to achieve robust, persistent, and recoverable long-horizon behaviors without modifying their parameters or requiring test-time adaptation. We further theoretically characterize the benefits of the harness-based approach and conduct extensive experiments across distribution shifts, long-horizon reasoning, and embodied-memory benchmarks. EVA achieves an improvement of **37.8%** over strong baselines while reducing token overhead by **10.9**$\times$ on VLABench. Real-world deployment on a humanoid robot further demonstrates its effectiveness. Overall, these results show that EVA can substantially extend the capabilities and generalization of existing embodied models. Code is available at `https://anonymous.4open.science/r/eva-0`.

1 INTRODUCTION

Recent advances in embodied foundation models (Intelligence et al., 2025; Kim et al., 2025; Intelligence et al., 2026b) have substantially improved the generality of robot control. Vision-Language-Action (VLA) models(Zitkovich et al., 2023; Intelligence et al., 2025; Kim et al., 2024; NVIDIA et al., 2025) and World Action Models (WAMs)(Ye et al., 2026; Zhou et al., 2026; Luo et al., 2026) can condition robot actions on visual observations and language instructions, enabling a wide range of manipulation skills. However, these models primarily operate at the level of action execution and still exhibit limited generalization across tasks (Zhang et al., 2025), environments, and often requiring repeated adaptation to new settings. Meanwhile, real-world instructions are often open-ended and long-horizon, such as "organize the kitchen" or "prepare breakfast", requiring robots to maintain task objectives, reason over intermediate goals with persistent memory, cope with noisy and changing environments, and recover from execution failures. However, consolidating all of these capabilities into a single VLA model to achieve reliable long-horizon task execution remains challenging.

Beyond VLA-based control, another line of work, such as Code-as-Policies (Liang et al., 2023; Fu et al., 2026), addresses these challenges by leveraging the strong coding capabilities of foundation models to compose and invoke low-level control primitives over multiple steps, demonstrating strong task-level generalization. However, code-based control can incur substantial generation and execution latency and struggle to express reactive, contact-rich behaviors that require fine-grained control trajectories, such as grasping irregularly shaped objects. Beyond code-based control, agent-based approaches (Wang et al., 2026; Zhang et al., 2026; Yang et al., 2025; Huo et al., 2026; Lei et al.,

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.

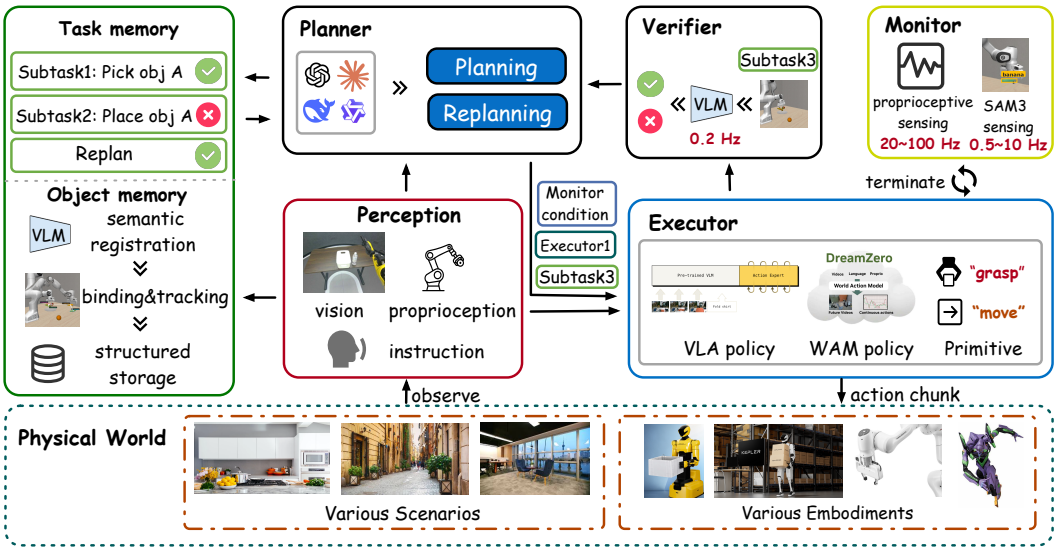

Figure 1: Overview of the EVA framework. Perception component collects visual, proprioceptive, and instructional signals from the physical world and organizes them into structured task and object Memory through semantic registration, binding, and tracking. Planner leverages the current memory state to decompose user instructions into executable subtasks and triggers replanning when necessary. Further, Executor supports multiple low-level execution mechanisms to transform subtasks into continuous action trajectories, while a high-frequency Monitor continuously tracks execution and low-frequency Verifier evaluates the outcome of each subtask. Execution outcomes are fed back into memory to support subsequent planning and recovery, forming a closed-loop system that operates across diverse scenarios and embodiments.

2026; Chen et al., 2026b) further introduce high-level reasoning around robot execution, including agents that guide underlying VLA policies. While these approaches begin to bridge task reasoning and robot control, they are often specifically tailored to the underlying policy model, task formulation, or execution mechanism (Lei et al., 2026; Peng et al., 2026; Li et al., 2026b). More recently, general-purpose code agents, such as Codex and Claude Code (Zhang et al., 2026), have also been explored for embodied tasks. However, generic agent architectures are not designed for continuous physical interaction and can incur substantial inference and token overhead. These limitations motivate a dedicated embodied harness, designed specifically to bridge general-purpose agentic reasoning with the persistent state, continuous monitoring, verification, and recovery required for long-horizon physical interaction.

To address this gap, we introduce **EVA**, a general and extensible embodied harness for long-horizon task execution. EVA decouples task-level orchestration from low-level action execution, allowing heterogeneous embodied policies and control primitives to serve as interchangeable executors. Given a user instruction, EVA continuously processes multimodal observations to extract task-relevant events, object states, and execution outcomes, which are consolidated into persistent task and object memory through updates. A memory-grounded Planner then accesses the relevant memory to generate the next executable subtask together with the appropriate executor and monitoring and verification specifications. During execution, EVA combines high-frequency monitoring with lower-frequency semantic verification, using the resulting feedback to update task and object memory and determine the next planning step or recovery action. This closed-loop architecture allows existing embodied models to be composed and orchestrated for long-horizon tasks without modifying or retraining their parameters.

Furthermore, we provide a theoretical analysis of harness-based method and evaluate EVA across diverse settings, including generalization, memory-dependent tasks, long-horizon manipulation, and real-world humanoid deployment. EVA achieves an improvement of 37.8% over strong baselines while reducing token overhead by 10.9× on VLABench. Without modifying underlying model parameters, EVA improves the reliability and effective task horizon of existing embodied models, highlighting its effectiveness as a general embodied harness.

2 Related Work

2.1 Embodied Foundation Models

The success of LLMs, VLMs, and other foundation models has motivated researchers to develop embodied foundation models, which aim to autonomously control robots based on language instructions, environmental observations, and robot states. Leveraging the strong image and language understanding capabilities of existing VLMs, researchers have proposed VLA models (Zitkovich et al., 2023; Black et al., 2026; Intelligence et al., 2025; Kim et al., 2024; NVIDIA et al., 2025). VLA models use VLMs to semantically interpret language instructions and environmental observations, and condition an action expert on the resulting multimodal features together with the robot state to generate actions. Since relying only on the current observation and robot state makes it difficult to maintain awareness of long-horizon tasks, some VLA models have begun to incorporate memory (Shi et al., 2026; Li et al., 2026a).

To better model changes in the environment, researchers have proposed World Models (Gao et al., 2026; Mur-Labadia et al., 2026) and WAMs(Ye et al., 2026; Zhou et al., 2026; Luo et al., 2026). Conditioned on the current world state, World Models predict the subsequent state of the world, while WAMs predict both the subsequent world state and robot actions. Recent works (Huang et al., 2026; Intelligence et al., 2026a) further use images predicted by World Models as goal images to guide VLA models in completing corresponding tasks, demonstrating that VLAs can benefit from World Models.

2.2 Embodied Agents

Although existing embodied foundation models can perform individual tasks, they are often not trained to switch between tasks, making it difficult for them to execute multiple tasks consecutively. Embodied harnesses provide a solution for adapting these foundation models to long-horizon tasks without further training them. Early efforts primarily attempted to align VLAs using stronger VLMs (Zhao et al., 2025; Jin & Zhang, 2026; Peng et al., 2026). To address the limitations of embodied foundation models in long-horizon task execution, embodied agents have gradually shifted toward task planning as their central capability, treating embodied foundation models as low-level execution tools (Xiao et al., 2026; Lei et al., 2026). Since long-horizon task execution requires agents to retain key information over extended interactions, researchers have introduced carefully designed memory mechanisms to address this challenge (Li et al., 2026b; Huo et al., 2026). More recently, general-purpose code agents such as Codex and Claude Code have also been explored for embodied tasks (Zhang et al., 2026). However, these architectures are not tailored to continuous physical interaction and may incur substantial inference and token overhead. Taken together, existing works provide different subsets of the components needed for robot, as summarized in Table 1.

Table 1: Comparison between EVA and existing embodied agents. We compare the architectural components supported by each system, including the Planner, Perception, Executor, Monitor, Verifier, Object Memory, and Task Memory.

Method	Planner	Perception	Executor	Monitor	Verifier	Object Memory	Task Memory
VLA^2 (Zhao et al., 2025)	✗	✓	✓	✗	✓	✓	✗
Agentic Robot (Xiao et al., 2026)	✓	✗	✓	✗	✓	✗	✓
ENPIRE (Xiao et al., 2026)	✓	✓	✓	✗	✓	✗	✗
RoboClaw (Li et al., 2026b)	✓	✗	✓	✗	✓	✗	✓
Harness VLA (Zhang et al., 2026)	✓	✓	✓	✗	✗	✗	✓
EVA	✓	✓	✓	✓	✓	✓	✓

3 Methodology

3.1 Overall Design

In this section, we introduce EVA, a general and extensible embodied agent harness that bridges high-level user instructions and physical execution. As shown in Figure 1, EVA consists of six functional components: *Perception*, *Memory*, *Planning*, *Execution*, *Monitoring*, and *Verification*, which together

provide a comprehensive interface spanning environment perception and task reasoning to robotic execution and outcome verification. We next describe these components in detail.

3.2 Perception and Memory

Reliable interaction with the physical world requires robot system to continuously perceive its surroundings and maintain a persistent representation of the information relevant to the ongoing task. In our framework, *Perception* and *Memory* form a tightly coupled interface between the physical world and downstream reasoning: perception converts continuously arriving multimodal observations into structured information, while memory organizes, persists, and updates this information to provide a consistent state for planning and execution.

Perception. EVA receives three types of signals during task execution: visual observations of the environment, proprioceptive states of the robot, and human instructions. These signals provide complementary information about the physical scene, the robot's current state, and the intended task, respectively. These multimodal signals are processed into structured semantic and state information, which can be further organized and integrated into other parts.

As for memory, we maintains two complementary forms of memory: *task memory* and *object memory*, which capture the evolving task context and the persistent state of the physical environment, respectively.

Task Memory. Task memory maintains the high-level context of the ongoing task, including the task objective, execution progress, completed actions, and relevant intermediate states. This information is continuously updated throughout task execution, allowing EVA to preserve long-horizon task context and avoid repeatedly inferring information that has already been established.

Object Memory. Object memory maintains a persistent, object-centric representation of the physical environment by grounding transient visual observations into consistent physical entities. EVA first employs a VLM for semantic registration, identifying objects in incoming images and extracting their semantic labels, attributes, and static spatial relations. However, VLMs are not explicitly designed for persistent and fine-grained object tracking. Therefore, EVA further introduces SAM3 for object binding and tracking: based on the semantic cues obtained from the VLM, SAM3 associates newly observed instances with their historical identities and tracks target objects across subsequent observations, thereby maintaining consistent object identities over time. The resulting object states are stored in a structured and compact object memory, providing temporally consistent environmental information for downstream planning and control.

3.3 Memory-Grounded Task Planning

Building on the persistent task and object memory maintained by the perception and memory modules, Planner then translates user instructions into executable subtasks grounded in the current task and environment state. In practice, user instructions can be ambiguous, underspecified, or too complex to execute directly, requiring the planner to infer the intended task structure and determine an appropriate sequence of subtasks. The Planner therefore decomposes high-level instructions into executable subtasks while continuously adapting the plan to the evolving environment state and the capabilities of downstream executors.

Specifically, let g denote the user instruction, and let M_t^o and M_t^τ denote the object memory and task memory at planning step t, respectively. The object memory M_t^o maintains structured information about relevant objects and their current states, while the task memory M_t^τ records the execution history and verification results. Given these sources of information, the planner produces the next executable subtask together with its execution requirements:

$$(z_t, e_t, c_t) = \mathcal{P}\left(g, M_t^o, M_t^\tau\right), \tag{1}$$

where z_t denotes the next executable subtask, e_t denotes the selected downstream executor for z_t, and c_t denotes an adaptive monitoring and verification specification associated with the subtask. The executor e_t is selected according to the requirements of z_t and the capabilities of the available downstream executors, while c_t specifies how and under what conditions the execution of z_t should be monitored and its outcome verified. These conditions are dynamically generated for each subtask and can be automatically instantiated or programmed by the agent.

Closed-loop Planning. EVA performs task planning in a closed-loop manner, generating only the next executable subtask z_t rather than committing to a complete sequence of future subtasks $z_{t+1}, z_{t+2}, \ldots, z_{t+n}$. This design is important because the environment is only partially observable and may change during execution, causing observations and object memory to evolve over time. Consequently, a plan generated entirely in advance may rely on outdated information and limit the planner's ability to adapt subsequent subtasks to the updated environment state.

Replanning and Recovery. Given the planning output (z_t, e_t, c_t), EVA dispatches the subtask z_t to the selected executor e_t and executes it under the monitoring and verification specification c_t. During execution, the Perception module collects environment observations o_t, while the Verifier evaluates the execution outcome and produces a completion status y_t. The resulting observations and execution feedback are then used to update the object and task memories:

$$M^o_{t+1} = \mathcal{U}_o\left(M^o_t, o_t\right), \qquad M^\tau_{t+1} = \mathcal{U}_\tau\left(M^\tau_t, z_t, y_t\right), \tag{2}$$

where $\mathcal{U}_o$ and $\mathcal{U}_\tau$ denote the object and task memory update operators, respectively, and o_t denotes the observations collected during the execution of z_t. Thus, M^o_{t+1} incorporates the latest environment state, while M^τ_{t+1} records the execution history and its verification outcome.

The updated memories $(M^o_{t+1}, M^\tau_{t+1})$, together with the completion status y_t, determine the next planning step. If y_t indicates successful completion, the Planner first checks whether the overall task specified by g has been completed. If not, it generates the next executable subtask and its execution requirements:

$$(z_{t+1}, e_{t+1}, c_{t+1}) = \mathcal{P}\left(g, M^o_{t+1}, M^\tau_{t+1}\right). \tag{3}$$

If y_t indicates failure, the Planner assesses whether recovery is feasible. When needed, it requests a return to a previously recorded stable checkpoint, which EVA selects and restores via a constrained joint-space trajectory. If recovery fails, the failure is returned to the Verifier and Planner for further recovery; otherwise, normal planning resumes. This closed-loop process repeats until g is satisfied or no feasible recovery is available.

3.4 Execution, Monitor and Verifier.

As discussed above, once the Planner generates a subtask, EVA executes it through a selected downstream executor while continuously monitoring its progress. The execution is terminated when the monitoring condition is met, after which the resulting outcome is verified and fed back to the Planner. Together, these components connect subtask-level planning with low-level robot execution and provide the execution feedback required for closed-loop planning.

Executor. The Executor serves as the operation interface between the agent and the physical world. EVA supports heterogeneous and extensible downstream executors, including programmatic methods and foundation-model-based policies. Programmatic methods use predefined algorithms, programs, or control procedures to generate executable trajectories, providing reliable execution for primitive tasks and robustness to variations such as object locations and colors, but limited flexibility for complex tasks. In contrast, foundation-model-based policies, such as VLAs and WAMs, can interpret task instructions and perform fine-grained manipulation, but may be less stable and degrade on states outside their training distributions. EVA therefore treats these mechanisms as executor candidates and selects e_t according to the requirements of z_t and the capabilities of the available executors in planner, enabling their complementary strengths to be combined within a unified planning loop.

Monitor and Verifier. Real-world robot execution is inherently noisy and uncertain, making closed-loop monitoring necessary for reliable task completion. Applying a semantic VLM verifier at every control step, however, would introduce substantial computation and latency. EVA therefore separate *rollout termination* from *semantic verification* using a high-frequency Monitor and a low-frequency triggered Verifier. The Monitor performs lightweight online monitoring and determines when the current rollout should terminate:

$$m_{t,k} = \text{Monitor}\left(c_t, o_{t,k}\right) \in \{continue, terminate\}, \tag{4}$$

where c_t denotes the termination conditions from planner, $o_{t,k}$ the current observations. Once the Monitor triggers termination, the robot stops and invoking the Verifier:

$$y_t = \text{Verifier}\left(z_t, o_{t,0}, o_{t,\rho_t}, M^o_{t+1}\right), y_t \in \{success, failure\}. \tag{5}$$

where ρ_t denotes the termination step. The Verifier aggregates multiple semantic and physical signals through weighted evidence to determine the execution outcome, which is then returned to the Planner for subsequent planning or recovery.

Overall, the execution loop integrates monitoring, verification, and replanning to support reliable closed-loop task execution. The following theoretical analysis isolates these components under simplified probabilistic assumptions.

4 Theoretical Analysis

In this section, we use a probabilistic analysis to characterize the benefits of our framework. We simplify our framework to focus on the executor, verifier, and recovery components for a single-step execution. Specifically, consider an atomic embodied task for which the executor succeeds in a single attempt with probability $p \in (0, 1)$. A binary verifier correctly identifies both successful and failed executions with probability $q \in (0, 1)$. When the verifier signals a failure, the system replans and retries the execution, up to m attempts. Thus, the probability of triggering a retry after a single execution is $r = p(1-q) + (1-p)q$. Based on this formulation, we derive the following proposition:

Proposition 4.1 (Benefit of harness)**.** *With at most m execution attempts, the probability of successfully completing the task is $S_m = pq(1 - r^m)/(1 - r)$. As the number of retries approaches infinity, the success probability converges to $S_\infty = pq/(1 - r)$. Moreover, the framework achieves a higher success probability than a standalone executor, i.e., $S_\infty > p$, if and only if $q > \frac{1}{2}$.*

This result shows that the verifier only needs to outperform random guessing to yield an asymptotic improvement in task-level success probability. In particular, it does not require a highly accurate verifier; even a moderately reliable verifier can improve overall task reliability.

Proposition 4.2 (Decay of retry gains)**.** *The marginal improvement for the m-th attempt is $\Delta S_m = S_m - S_{m-1} = pqr^{m-1}$. Since $0 < r < 1$, the marginal gains decay geometrically, with $\Delta S_{m+1}/\Delta S_m = r$.*

The proposition indicates that early retries provide the largest improvement, while the incremental benefit of additional attempts becomes progressively smaller. Tasks with lower base success probability or less reliable verifier benefit more from larger retry budgets.

Proposition 4.3 (Execution-monitor trade-off)**.** *When $p + q > 1$, the marginal benefit of improving the component with higher accuracy is larger. Specifically,*

$$\frac{\partial S_\infty}{\partial p} - \frac{\partial S_\infty}{\partial q} = \frac{(q-p)(1-p-q)}{(1-p-q+2pq)^2}.$$

Thus, if $p > q$, improving execution accuracy yields a larger marginal gain, while if $q > p$, improving verifier accuracy yields a larger marginal gain.

This result suggests prioritizing improvements to the currently stronger component when the executor and verifier are jointly effective.

Proposition 4.4 (Performance under imperfect reset)**.** *Suppose the recovery operation succeeds with probability $k \in (0, 1)$. The framework outperforms standalone execution if and only if $q > (1 - pk)/(1 + k - 2pk) > 1/2$. Thus, imperfect recovery increases the verifier accuracy required for the framework to achieve a net performance gain.*

This suggests that, in real-world settings where recovery is imperfect, the framework requires a more accurate verifier to achieve a net performance gain.

5 Experiments

In this section, we evaluate EVA in simulation benchmarks and on a real-world long-horizon task to answer the following key questions:

- RQ1: Does EVA improve robustness and generalization on manipulation tasks?
- RQ2: Does EVA exhibit strong capabilities on memory-dependent tasks?

- RQ3: Does EVA improve performance on long-horizon composite tasks?
- RQ4: How well does EVA perform in real-world long-horizon scenarios?

Further experiments, including ablation studies, scaling analysis, and timeline analysis, are detailed in Appendices C and D.

5.1 Experimental Setup

Simulation Benchmarks. To comprehensively evaluate the effectiveness of EVA, we evaluate robotic capabilities across three categories: *generalizability*, *memory management*, and *long-horizon reasoning*. For *generalizability*, we evaluate (1) VLABench (Zhang et al., 2025), a large-scale benchmark covering texture, semantic, appearance shifts across five official tracks. For *memory management*, we evaluate (2) RoboMME (Dai et al., 2026), covering Counting, Permanence, Reference, and Imitation. For *long-horizon reasoning*, we evaluate (3) LIBERO-LoHo (Huang et al., 2026), a long-horizon benchmark derived from the standard LIBERO benchmark (Liu et al., 2023), which involves five to seven dependent manipulation steps and requires composition of primitive skills. Detail statistics and settings for benchmarks are shown in Appendix C.

Baselines. To enable comprehensive comparison, we compare our proposed model with representative baselines across two categories: VLA policies, and agentic systems for robotics. For agentic systems, we compare with Harness VLA (Zhang et al., 2026) and VoLo (Chen et al., 2026a). For all agentic systems and their underlying VLA baselines, we conduct our own evaluations to obtain performance metrics. Results for other baselines are taken from their corresponding benchmark reports or prior published evaluations.

Implementation Details. Unless otherwise specified, all VLA baselines are evaluated using publicly released checkpoints and recommended evaluation protocols. For all agentic-system methods, we instantiate them with Xiaomi-Robotics-1 (denoted π_{XR1}) or $\pi_{0.5}$ on VLABench, with the RLinf-released (Yu et al., 2026) $\pi_{0.5}$ SFT checkpoint for full-shot LIBERO-130 (denoted π_{RLinf}) on LIBERO-LoHo, and with the GroundSG variant of $\pi_{0.5}$ on RoboMME. We use Deepseek-V4.1-Flash (Xu et al., 2026) as the default language model for planning and Qwen3.8-Max (Qwen Team, 2026) for verifying. We evaluate all methods without test-time adaptation: the internal state is held fixed by resetting the configuration every episode, with no exploratory episodes conducted on the evaluation split. Detailed settings are provided in the Appendix C.

We also conduct real-world experiments on Kepler K2 humanoid robot, which is equipped with 42 degrees of freedom. For perception part, we mount a Gemini330 camera on the robot's head and two Intel RealSense D405 cameras on the wrists. This setup provides both egocentric view and two wrist-level visual observations for manipulation. For the humanoid VLA backbone, we adopt Ψ_0(Wei et al., 2026) as the underlying policy. We collect real-world demonstrations using a PICO 4 Ultra headset with XRoboToolkit (Zhao et al., 2026) teleoperation. In total, we collect 204 demonstration trajectories, comprising 282,548 decision steps for fine-tuning phase. All demonstrations are collected under a unified control frequency of 30 Hz. The deployed humanoild VLA are executed on NVIDIA Blackwell 6000 GPU. Details of deployment are shown in the Appendix F.

5.2 Experiment Results

Manipulation Generalization (RQ1). We first evaluate standard and perturbed manipulation. VLABench tests generalization across semantic, visual, and instruction shifts. As shown in Table 2, The results demonstrate that EVA consistently improves manipulation generalization across different VLA backbones without modifying their underlying parameters. For π_{XR1}, EVA improves the average SR, PS, and IS by 7.4%, 7.2%, and 4.0%, respectively, while for $\pi_{0.5}$, the corresponding metrics improve by 13.8%, 14.1%, and 6.8%, respectively. These results demonstrate that EVA provides consistent gains across different VLA backbones. Compared with other agentic methods, EVA (π_{XR1}) achieves higher average SR, PS, and IS, while reducing the average latency and token consumption to 171.0 s and 70.2K, respectively, demonstrating improved task generalization and inference efficiency.

Table 2: Results on VLABench. We report success rate (SR), progress score (PS), intention score (IS), and the average latency and token consumption of successful episodes, where PS measures the progressive task completion and IS measures the fraction of intended target objects correctly identified. The best and second-best results are highlighted in **bold** and underline, respectively.

Method	In-dist.			Cross Category			Commonsense			Instruction			Texture			Average			Latency (s) ↓	Tokens (K) ↓
	SR ↑	PS ↑	IS ↑	SR ↑	PS ↑	IS ↑	SR ↑	PS ↑	IS ↑	SR ↑	PS ↑	IS ↑	SR ↑	PS ↑	IS ↑	SR ↑	PS ↑	IS ↑		
π_{XR1}	79.0	87.9	79.8	57.0	69.7	68.4	61.0	66.5	65.7	67.0	74.2	73.0	65.0	76.6	77.8	65.8	75.0	72.9	–	–
$\pi_{0.5}$	43.0	61.1	75.8	26.0	42.9	70.7	26.0	45.4	66.3	21.0	40.3	67.7	31.0	49.3	**76.7**	29.4	47.8	70.2	–	–
ACoT-VLA	–	66.1	79.8	–	38.9	54.1	–	37.8	52.3	–	39.6	56.8	–	54.6	74.6	–	47.4	63.5	–	–
ERVLA	69.7	81.1	**84.2**	47.0	61.0	66.4	44.0	55.0	57.2	58.0	70.2	73.8	47.4	62.3	70.6	53.2	65.9	70.4	–	–
Harness VLA	41.0	61.2	81.0	34.0	52.1	**73.4**	32.0	46.3	65.0	38.0	56.2	76.0	32.0	52.5	75.0	35.4	53.7	74.1	341.9	766.5
EVA ($\pi_{0.5}$)	63.0	76.1	83.8	38.0	56.2	68.0	43.0	61.6	**78.8**	40.0	62.4	**84.5**	32.0	53.4	69.7	43.2	61.9	**77.0**	198.2	96.6
EVA (π_{XR1})	**84.0**	**91.7**	79.6	**65.0**	**74.0**	71.0	**77.0**	**83.2**	78.0	**76.0**	**86.5**	81.0	**64.0**	**75.7**	75.0	**73.2**	**82.2**	76.9	**171.0**	**70.2**

Memory Utilization (RQ2). We evaluate our method on RoboMME, which comprises four task suites (Counting, Permanence, Reference, and Imitation), under the *Memory-Agent* setting, where an external agent generates grounded subgoals associated with image coordinates to guide policy execution. As shown in Table 3, EVA achieves 61.25%, outperforming other baseline by 55.53%. While conventional GroundSG-based symbolic memory performs well on Counting, it struggles with tracking and trajectory reproduction, where abstract representations may lose fine-grained visual. In contrast, EVA preserves these cues while retaining the benefits of symbolic temporal reasoning, leading to substantial gains on Permanence and Imitation. We further evaluate EVA under the *Memory-QA* setting in Appendix D.

Table 3: Main results on RoboMME. The first column combines the subgoal source and subgoal representation. The four column blocks correspond to the benchmark task suites.

Method	**Counting**				**Permanence**				**Reference**				**Imitation**				**AVG**
	Bin Fill	Pick Xtimes	Swing Xtimes	Stop Cube	Video Umsk	Button Umsk	Video UmskS	Button UmskS	Pick HighL	Video Repick	Video PlcBtn	Video PlcOrd	Move Cube	Insert Peg	Pattern Lock	Route Stick	
Deepseek V4.1 + GroundSG	12.50	20.00	20.00	20.00	33.33	11.11	50.00	0.00	20.00	28.57	12.50	14.29	62.50	0.00	20.00	12.50	19.83
Qwen3.8-Max + GroundSG	12.50	55.56	0.00	0.00	71.43	10.00	50.00	0.00	37.50	0.00	0.00	0.00	70.00	0.00	11.11	0.00	19.88
Harness VLA + GroundSG	20.00	70.00	0.00	10.00	70.00	50.00	40.00	30.00	80.00	10.00	50.00	80.00	80.00	0.00	40.00	0.00	39.38
EVA + GroundSG	100.00	100.00	0.00	0.00	100.00	100.00	80.00	0.00	100.00	50.00	100.00	100.00	100.00	0.00	50.00	0.00	61.25

Long-horizon Decomposition (RQ3). We evaluate EVA on LIBERO-LoHo, a challenging long-horizon benchmark with task horizons of large steps, substantially longer than the 1–3-step tasks in standard LIBERO. As shown in Table 4, EVA achieves 79.0% success rate, outperforming the strongest baseline by 14.2%. The improvement is particularly pronounced on longer and more dependency-heavy tasks, where successful execution requires maintaining consistency across multiple intermediate actions. In contrast, standard VLA policies often achieve partial task progress but fail to complete the full sequence, highlighting the difficulty of compounding errors over long horizons. These results demonstrate that EVA provides effective long-horizon reasoning and execution beyond the capabilities of the underlying VLA policy.

Real-world Experiment (RQ4). We evaluate EVA on a long-horizon manipulation task deployed on a physical humanoid robot, comparing direct execution of a humanoid VLA policy with the same policy augmented with EVA. Both methods use the same VLA backbone trained on the same set of real-world demonstrations, isolating the contribution of the agentic harness. Our task requires the robot to put several bottles into a box in order. Otherwise, it may knock over another bottle while transporting one bottle. The task requires the humanoid robot to complete a sequence of manipulation subgoals over an extended horizon, involving visual perception, bimanual coordination, and sustained interaction with the environment.

Figure 2 demonstrates the performance of EVA on the Kepler K2 humanoid robot. Under VLA-only planning, the given instruction causes the VLA to first attempt the high-risk grasp of the white bottle, which results in an unintended collision with the orange bottle and ultimately prevents the task from being completed. In contrast, EVA correctly plans to place the orange bottle before the white bottle, thereby avoiding the unintended collision.

Table 4: Results on LIBERO-LoHo. We report Q-Score (QS, %) and task success rate (SR, %), and the average latency and token consumption on successful episodes, where QS denotes the fraction of completed sub-goals relative to all sub-goals, and SR measures the percentage of tasks fully completed. The best and second-best results are highlighted in **bold** and underline, respectively.

Method	Task 1		Task 2		Task 3		Task 4		Task 5		Average		Latency (s) ↓	Tokens (K) ↓
	QS ↑	SR ↑	QS ↑	SR ↑	QS ↑	SR ↑	QS ↑	SR ↑	QS ↑	SR ↑	QS ↑	SR ↑		
H-WM-$\pi_{0.5}$	**98.0**	**94.0**	86.7	**60.0**	**74.0**	**46.0**	70.7	42.0	**95.0**	**82.0**	**84.9**	**64.8**	–	–
H-WM-Stable-Diffusion-$\pi_{0.5}$	92.7	82.0	80.0	46.0	65.3	30.0	74.0	34.0	93.5	80.0	81.1	54.4	–	–
Logic-guided $\pi_{0.5}$	95.3	86.0	84.7	58.0	54.7	16.0	39.3	4.0	92.0	78.0	73.2	48.4	–	–
LLM-guided $\pi_{0.5}$	84.7	54.0	80.7	42.0	68.0	24.0	41.3	4.0	59.5	10.0	66.8	26.8	–	–
$\pi_{0.5}$	66.0	4.0	73.3	24.0	54.7	4.0	44.7	0.0	38.0	0.0	55.3	6.4	–	–
π_0	54.0	0.0	62.0	28.0	44.0	0.0	31.3	0.0	34.0	0.0	45.1	5.6	–	–
X-VLA	48.8	0.0	16.7	0.0	23.3	0.0	33.3	0.0	40.0	0.0	32.4	0.0	–	–
Gr00t-N1.5	0.0	0.0	30.7	0.0	34.7	0.0	2.7	0.0	0.0	0.0	13.6	0.0	–	–
OpenVLA-OFT	0.0	0.0	60.0	0.0	17.3	0.0	22.7	0.0	0.0	0.0	20.0	0.0	–	–
OpenVLA	0.0	0.0	2.7	0.0	24.0	0.0	2.7	0.0	0.0	0.0	5.9	0.0	–	–
π_{RLinf}	65.0	0.0	68.3	10.0	55.0	5.0	43.3	0.0	43.8	0.0	55.1	3.0	–	–
Harness VLA	52.0	5.0	74.2	30.0	84.2	55.0	**89.2**	**80.0**	52.9	0.0	70.5	34.0	**539.0**	6066.4
VoLo	75.4	50.0	80.7	50.0	61.7	25.0	58.3	10.0	27.5	0.0	60.7	27.0	1718.4	127.7
EVA (π_{RLinf})	96.7	90.0	**88.3**	**75.0**	**90.0**	**75.0**	**100.0**	**100.0**	81.2	55.0	**91.2**	**79.0**	649.5	**97.7**

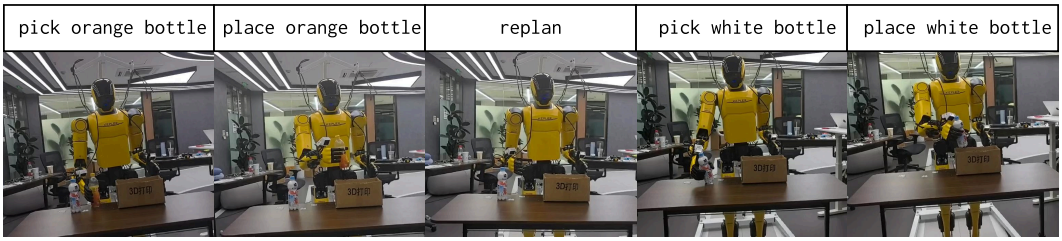

Figure 2: Real world deployment of EVA on Kepler K2 robot.

6 Conclusion

We present EVA, a memory-augmented harness for long-horizon embodied tasks that decouples high-level task orchestration from low-level action models. By integrating task- and object-centric memory with closed-loop planning, execution, monitoring, verification, and recovery, EVA extends existing VLA models without modifying or retraining their parameters. Experiments across memory-intensive tasks, long-horizon benchmarks, and real-world humanoid deployment show consistent gains in reliability and long-horizon capabilities, including a 37.8% improvement over strong baselines on VLABench while reducing token overhead by 10.9×. These results demonstrate the significant potential of embodied harnesses for extending the effective capabilities of existing foundation models, and we hope EVA will establish harness-based orchestration as a foundational building block for real-world robotic agents, paving the way toward generalist robots capable of reliably completing open-ended physical tasks.

AI USE STATEMENT

In this work, we used generative AI tools solely to assist with polishing and improving the clarity, grammar, and presentation of the manuscript. We did not use generative AI tools to generate research ideas, design experiments, or produce research results. All text has been carefully reviewed and verified by the authors for correctness, consistency, and fidelity to the underlying work. The authors take full responsibility for the final content of this work, including all claims, statements, and artifacts.

REPRODUCIBILITY STATEMENT

We have made every effort to ensure that both the theoretical results and experimental findings in this work are fully reproducible. For the theoretical analysis, all assumptions and detailed proofs are provided in the Appendix. For the experimental results, we provide complete implementation details, training and evaluation protocols, and relevant experimental settings in the main paper and appendix, together with the source code and supplementary materials. These materials are intended to provide sufficient information for independently reproducing our theoretical results and empirical findings.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2026. URL https://arxiv.org/abs/2410.24164.

Siyi Chen, Hugo Hadfield, Alex Zook, Mikaela Angelina Uy, Chan Hee Song, Erwin Coumans, Xuning Yang, Faisal Ladhak, Qing Qu, Stan Birchfield, et al. Volo: A physical orchestrator for open-vocabulary long-horizon manipulation. *arXiv preprint arXiv:2606.07723*, 2026a.

Yanzhe Chen, Zechen Bai, Zhijun Cao, Wenzheng Zeng, Kevin Qinghong Lin, Yiqi Lin, Guoqiang Liang, Kevin Yuchen Ma, Qiming Huang, and Mike Zheng Shou. Show-harness: Just a vlm agent can play robots, 2026b. URL https://arxiv.org/abs/2609.10522.

Yinpei Dai, Hongze Fu, Jayjun Lee, Yuejiang Liu, Haoran Zhang, Jianing Yang, Chelsea Finn, Nima Fazeli, and Joyce Chai. Robomme: Benchmarking and understanding memory for robotic generalist policies. *arXiv preprint arXiv:2603.04639*, 2026.

Letian Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Guanzhi Wang, Dantong Niu, Fei-Fei Li, et al. Cap-x: A framework for benchmarking and improving coding agents for robot manipulation. *arXiv preprint arXiv:2603.22435*, 2026.

Shenyuan Gao, William Liang, Kaiyuan Zheng, Ayaan Malik, Seonghyeon Ye, Sihyun Yu, Wei-Cheng Tseng, Yuzhu Dong, Kaichun Mo, Chen-Hsuan Lin, Qianli Ma, Seungjun Nah, Loic Magne, Jiannan Xiang, Yuqi Xie, Ruijie Zheng, Dantong Niu, You Liang Tan, K. R. Zentner, George Kurian, Suneel Indupuru, Pooya Jannaty, Jinwei Gu, Jun Zhang, Jitendra Malik, Pieter Abbeel, Ming-Yu Liu, Yuke Zhu, Joel Jang, and Linxi "Jim" Fan. Dreamdojo: A generalist robot world model from large-scale human videos, 2026. URL https://arxiv.org/abs/2602.06949.

Jinbang Huang, Wenyuan Chen, Zhiyuan Li, Oscar Pang, Xiao Hu, Lingfeng Zhang, Yuanzhao Hu, Zhanguang Zhang, Mark Coates, Tongtong Cao, et al. H-wm: Robotic task and motion planning guided by hierarchical world model. *arXiv preprint arXiv:2602.11291*, 2026.

Dongjie Huo, Haoyun Liu, Guoqing Liu, Dekang Qi, Zhiming Sun, Maoguo Gao, Jianxin He, Yandan Yang, Xinyuan Chang, Feng Xiong, Xing Wei, Zhiheng Ma, and Mu Xu. Abot-claw: A foundation for persistent, cooperative, and self-evolving robotic agents, 2026. URL https://arxiv.org/abs/2604.10096.

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.

Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, Vedant Choudhary, Foster Collins, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Maitrayee Dhaka, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachlan Groom, Haroun Habeeb, Hunter Hancock, Karol Hausman, Gashon Hussein, Victor Hwang, Brian Ichter, Connor Jacobsen, Szymon Jakubczak, Rowan Jen, Tim Jones, Gregg Kammerer, Ben Katz, Liyiming Ke, Mairbek Khadikov, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Brendon LeCount, Sergey Levine, Xinyu Li, Adrian Li-Bell, Vladislav Lialin, Zhonglin Liang, Wallace Lim, Yao Lu, Enyu Luo, Vishnu Mano, Nandan Marwaha, Aikys Mongush, Liam Murphy, Suraj Nair, Tyler Patterson, Karl Pertsch, Allen Z. Ren, Gavin Schelske, Charvi Sharma, Baifeng Shi, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Jiaming Tang, Jimmy Tanner, Shalom Tekeste, Marcel Torne, Kyle Vedder, Quan Vuong, Anna Walling, Haohuan Wang, Jason Wang, XuDong Wang, Chris Whalen, Samuel Whitmore, Blake Williams, Charles Xu, Sukwon Yoo, Lili Yu, Wuming Zhang, Zhuoyang Zhang, and Ury Zhilinsky. $\pi_{0.7}$: a steerable generalist robotic foundation model with emergent capabilities, 2026a. URL https://arxiv.org/abs/2604.15483.

Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, et al. $\pi_{0.7}$: a steerable generalist robotic foundation model with emergent capabilities. *arXiv preprint arXiv:2604.15483*, 2026b.

Ruofan Jin and Zaixi Zhang. Agentic-vla: Efficient online adaptation for vision-language-action models, 2026. URL https://arxiv.org/abs/2605.22896.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

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

Zixing Lei, Changxing Liu, Yichen Xiong, Minhao Xiong, Yuanzhuo Ding, Zhipeng Zhang, Weixin Li, and Siheng Chen. Towards long-horizon embodied agents with tool-aligned vision-language-action models, 2026. URL https://arxiv.org/abs/2605.13119.

Peiyan Li, Yuze Zhu, Yixiang Chen, Qisen Ma, Yuan Xu, Jiabing Yang, He Guan, Yan Huang, Hongtao Wu, Xiao Ma, Tao Kong, Liang Wang, and Tieniu Tan. Bridgevla++: A data-efficient, generalizable, and memory-augmented vision-language-action framework for 3d manipulation, 2026a. URL https://arxiv.org/abs/2608.05042.

Ruiying Li, Yunlang Zhou, YuYao Zhu, Kylin Chen, Jingyuan Wang, Sukai Wang, Kongtao Hu, Minhui Yu, Bowen Jiang, Zhan Su, et al. Roboclaw: An agentic framework for scalable long-horizon robotic tasks. *arXiv preprint arXiv:2603.11558*, 2026b.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International conference on robotics and automation (ICRA)*, pp. 9493–9500. IEEE, 2023.

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.

Hao Luo, Wanpeng Zhang, Yicheng Feng, Sipeng Zheng, Haiweng Xu, Chaoyi Xu, Ziheng Xi, Yuhui Fu, and Zongqing Lu. Being-h0.7: A latent world-action model from egocentric videos, 2026. URL https://arxiv.org/abs/2605.00078.

Lorenzo Mur-Labadia, Matthew Muckley, Amir Bar, Mido Assran, Koustuv Sinha, Mike Rabbat, Yann LeCun, Nicolas Ballas, and Adrien Bardes. V-jepa 2.1: Unlocking dense features in video self-supervised learning, 2026. URL https://arxiv.org/abs/2603.14482.

NVIDIA, Johan Bjorck, Nikita Cherniadev Fernando Castañeda, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots. In *ArXiv Preprint*, March 2025.

Jiaqi Peng, Xiqian Yu, Delin Feng, Yuqiang Yang, Wenzhe Cai, Jing Xiong, Ganlin Yang, Jinliang Zheng, Jiafei Cao, Xueyuan Wei, et al. Cortex: A bidirectionally aligned embodied agent framework for long-horizon manipulation. *arXiv preprint arXiv:2607.05377*, 2026.

Qwen Team. Qwen3.8-max: A new bar for coding and cowork, August 2026. URL https://qwen.ai/blog?id=qwen3.8.

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, 2026. URL https://arxiv.org/abs/2508.19236.

Qi Wang, Tianyi Wang, Chengyang Li, Shikun Ban, Yurun Chen, Yizhong Ge, Jason Qin, Chengtai Li, and Wentao Zhu. Towards the harness of embodied agents, 2026. Technical Report.

Songlin Wei, Hongyi Jing, Boqian Li, Zhenyu Zhao, Jiageng Mao, Zhenhao Ni, Sicheng He, Jie Liu, Xiawei Liu, Kaidi Kang, et al. ψ_0: An open foundation model towards universal humanoid loco-manipulation. *arXiv preprint arXiv:2603.12263*, 2026.

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

Anyi Xu, B Li, Bangcai Lin, Bing Xue, BingCheng Xian, Bingzheng Xu, Bochao Wu, Bowei Zhang, Boyi Deng, CC Yu, et al. Deepseek-v4. 1-flash: Pushing the limits of kv cache compression. *arXiv preprint arXiv:2609.19969*, 2026.

Zhejian Yang, Yongchao Chen, Xueyang Zhou, Jiangyue Yan, Dingjie Song, Yinuo Liu, Yuting Li, Yu Zhang, Pan Zhou, Hechang Chen, and Lichao Sun. Agentic robot: A brain-inspired framework for vision-language-action models in embodied agents, 2025. URL https://arxiv.org/abs/2505.23450.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies, 2026. URL https://arxiv.org/abs/2602.15922.

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. In *20th USENIX Symposium on Operating Systems Design and Implementation (OSDI 26)*, pp. 829–846, 2026.

Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, et al. Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11142–11152. IEEE, 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, 2026. URL https://arxiv.org/abs/2607.08448.

Han Zhao, Jiaxuan Zhang, Wenxuan Song, Pengxiang Ding, and Donglin Wang. vla^2: Empowering vision-language-action models with an agentic framework for unseen concept manipulation, 2025. URL https://arxiv.org/abs/2510.14902.

Zhigen Zhao, Liuchuan Yu, Ke Jing, and Ning Yang. Xrobotoolkit: A cross-platform framework for robot teleoperation. In *2026 IEEE/SICE International Symposium on System Integration (SII)*, pp. 15–20. IEEE, 2026.

Pengfei Zhou, Shengcong Chen, Di Chen, Jiaxu Wang, Rongjun Jin, Bingwen Zhu, Yike Pan, Songen Gu, Kuanning Wang, Shufeng Nan, Xingyu Qiu, Chenhao Qiu, Pu Yang, Yunuo Cai, Jianxiong Gao, Yifan Li, Yanwei Fu, Xiangyu Yue, Zhi Chen, and Jianlan Luo. τ_0-wm: A unified video-action world model for robotic manipulation, 2026. URL https://arxiv.org/abs/2606.01027.

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 *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=XMQgwiJ7KSX.

A NOTATIONS

The main notations are summarized in Table 5.

Table 5: Description of major notations used in the EVA framework and its theoretical analysis.

Notation	Description
g	The user instruction or overall task goal.
M_t^o, M_t^τ	Object memory and task memory at planning step t.
z_t, e_t, c_t	The selected subtask, executor, and associated monitoring/verification conditions at step t.
$\mathcal{P}$	Planner that maps the instruction and current memories to (z_t, e_t, c_t).
$\mathcal{M}, \mathcal{V}$	Online monitor and outcome verifier, respectively.
ρ_t, y_t	Monitor termination signal and verification outcome for subtask z_t.
p, q	Single-attempt execution success probability and monitor accuracy, respectively.
r	Probability that one execution triggers a retry, $r = p(1-q) + (1-p)q$.
m	Maximum number of execution attempts considered in the theoretical analysis.
S_m, S_∞	Task success probability with at most m attempts and with unlimited retries, respectively.
ΔS_m	Marginal success-probability gain from allowing the m-th attempt.
k	Reset success probability in the imperfect-reset analysis.

B FRAMEWORK

The pseudocode of the algorithm underlying EVA is presented in Algorithm 1.

Algorithm 1 EVA: Memory-Grounded Closed-Loop Task Execution

Require: User instruction g and initial observation o_0
Ensure: Task execution outcome

$M_0^o \leftarrow \mathcal{U}_o(\emptyset, o_0)$ {Initialize object memory from the initial observation}
$M_0^\tau \leftarrow \mathrm{InitTaskMemory}(g)$ {Initialize task memory with the user instruction}
while true do
  $(z_t, e_t, c_t) \leftarrow \mathcal{P}(g, M_t^o, M_t^\tau)$ {z_t: subtask; e_t: execution tool; c_t: monitoring conditions}
  if $z_t = \varnothing$ **then**
    break {No further subtask is generated}
  end if
  Execute z_t using e_t under c_t {Executor performs the current subtask}
  $\rho_t \leftarrow \mathcal{M}(c_t, \{o_{t,k}\}_{k=1}^K)$ {Monitor determines when to terminate the rollout}
  $M_{t+1}^o \leftarrow \mathcal{U}_o(M_t^o, o_t)$ {o_t: observations collected during execution}
  $y_t \leftarrow \mathcal{V}(z_t, \rho_t, M_{t+1}^o)$ {Verifier evaluates the terminated execution}
  $M_{t+1}^\tau \leftarrow \mathcal{U}_\tau(M_t^\tau, z_t, y_t)$ {Update task memory with the verification result}
  $t \leftarrow t + 1$ {Replan using the updated memories}
end while
return Task execution outcome

C Additional Experimental Setup

Details on Simulation Benchmarks. In the following, we provide descriptions of the datasets used in our experiments, along with their corresponding statistics and settings.

- **VLABench.** VLABench evaluates language-conditioned manipulation across five official tracks: In-distribution, Cross-Category, Commonsense, Instruction, and Texture. Each track contains 10 evaluation tasks with 50 episodes per task. Due to the computational cost, we uniformly sample 10 episodes from each official 50-episode track-task set using NumPy Generator (PCG64) with seed 0, resulting in 500 rollouts per method. We report Success Rate (SR), Progress Score (PS), and Intention Score (IS).
- **RoboMME.** RoboMME evaluates memory-dependent manipulation across four task suites: Counting, Permanence, Reference, and Imitation. Each suite contains 4 tasks with 50 episodes per task. We uniformly sample 10 episodes from each official 50-episode task set, resulting in 160 rollouts per method.
- **LIBERO-LoHo.** LIBERO-LoHo extends LIBERO into five long-horizon tasks with five to seven dependent manipulation steps. We evaluate each LIBERO-LoHo task on benchmark-provided initial states 0–19 with seed 0, resulting in 100 rollouts per method.

Table 6: Statistics of simulation benchmarks used in our experiment.

Dataset	Tasks	Episodes / Task	Total Rollouts	Evaluation Metrics	Challenges
VLABench	50	10	500	SR, PS, IS	Distribution shifts
RoboMME	16	10	160	SR	Memory dependency
LIBERO-LoHo	5	20	100	SR, QS	Long-horizon composition

Setup for Ablation Studies. To assess the contribution of each module in EVA, we ablate the main modules of EVA on LIBERO-LoHo and RoboMME while keeping the evaluation protocol fixed. The ablation uses the same metrics as the main LIBERO-LoHo and RoboMME experiments. Ablation studies on LIBERO-LoHo are conducted on benchmark-provided initial states 0–9 with seed 0, resulting in 50 rollouts per variant, while ablations on RoboMME follow our experimental setup described above. In the following, we demonstrate the implementation of each variant of EVA used in ablation studies.

- **EVA (w/o monitor).** This variant disables all executor termination rules except the timeout rule, which terminates execution upon reaching the maximum step limit.
- **EVA (w/o verifier).** This variant disables the whole verifier module. It proceeds to the next planning step once the current step terminates.
- **EVA (w/o recovery).** This variant disables the recovery function that restores the robot and environment to a stable state for subsequent execution. Consequently, all subsequent executions begin from the current state only.
- **EVA (w/o memory).** This variant removes the object, task memory modules, clearing the object memory container and discarding all previous context regarding execution progress, completed actions, and relevant intermediate states.

D Additional Experimental Results

Ablation Studies. To demonstrate the effectiveness of the main modules of EVA, we ablate the monitor, verifier, recovery modules on LIBERO-LoHo and the object, task memory modules on RoboMME.

As shown in Table 7, removing the monitor, verifier, recovery module from EVA results in a 38.0%, 8.0%, 46.0% drop in success rate, respectively. EVA without the monitor module fails to terminate the VLA executor promptly after executions, leading to subsequent undesired executor behavior. Without the verifier, tracking of the current task stage becomes less reliable, leading to erroneous planning.

Without the recovery module, subsequent VLA executions in some cases start from out-of-distribution states, resulting in degraded performance. As shown in Table 8, removing the memory module results in a significant drop, indicating that the effectiveness of EVA.

Table 7: Results of ablation studies on LIBERO-LoHo.

Variant	QS (%) ↑	SR (%) ↑	Latency (s) ↓	Tokens (K) ↓
EVA	**91.7**	**78.0**	622.3	104.2
EVA (w/o monitor)	76.5	40.0	479.0	**50.4**
EVA (w/o verifier)	88.7	70.0	**247.4**	89.8
EVA (w/o recovery)	66.2	32.0	640.3	115.1

Table 8: Results of ablation studies on RoboMME.

Method	Counting				Permanence				Reference				Imitation				AVG
	Bin Fill	Pick Xtimes	Swing Xtimes	Stop Cube	Video Umsk	Button Umsk	Video UmskS	Button UmskS	Pick HighL	Video Repick	Video PlcBtn	Video PlcOrd	Move Cube	Insert Peg	Pattern Lock	Route Stick	
EVA (w/o memory)	0.00	40.00	0.00	10.00	30.00	10.00	10.00	0.00	80.00	10.00	50.00	80.00	10.00	0.00	0.00	0.00	20.63
EVA	100.00	100.00	0.00	0.00	100.00	100.00	80.00	0.00	100.00	50.00	100.00	100.00	100.00	0.00	50.00	0.00	61.25

Scaling with Inference Budget. We further evaluate whether EVA can effectively exploit additional inference time when coupled with a VLA policy. As shown in Figures 3 and 4, increasing environment steps for pure VLA policies barely improve SR, while increasing the inference budget for EVA consistently improves the cumulative success rate: the agent continues to solve additional tasks as more budget is allocated for reasoning. Performance rises rapidly at lower budgets and gradually saturates as the inference budget increases. This trend is consistent with Theorem 4.2, which predicts progressively smaller marginal gains from additional reasoning.

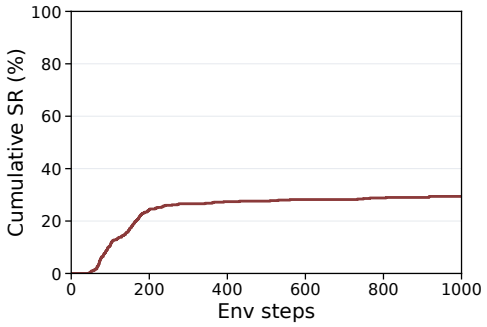

Figure 3: Cumulative success rate vs. Environment steps of $\pi_{0.5}$ policy on VLABench.

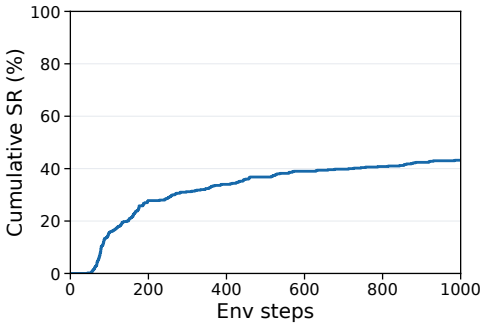

Figure 4: Cumulative success rate vs. Environment steps of EVA ($\pi_{0.5}$) on VLABench.

Deployment Timeline Analysis. To understand the practical cost of deploying EVA, we further analyze the end-to-end execution timeline of EVA on a LIBERO-LoHo task. The Gantt chart in Figure 5 decomposes each task execution into several time-consuming stages, including planning, recovery, VLA inference, verification and perception. This temporal breakdown provides a direct view of how the additional components introduced by the Harness contribute to the overall execution latency.

The timeline highlights how EVA allocates computation across the execution loop. VLA inference intervals remain short, lasting only a few seconds per control segment, while most additional latency appears between execution slots. Thus, the harness adds limited cost to individual VLA actions and concentrates its overhead on subtask transitions.

This coordination is event-driven rather than periodic. The monitor triggers verification only after detecting a termination condition, after which the system either proceeds, replans, or performs recovery. Unlike fixed-frequency polling, this design avoids repeatedly invoking verification during

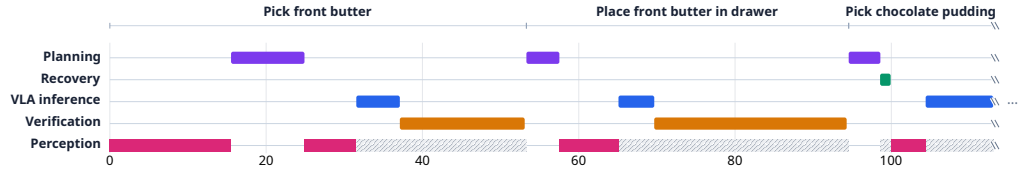

Figure 5: Gantt chart of the deployment timeline. Solid segments denote blocking synchronization gates, while hatched segments denote asynchronous perception overlapping with other stages. The remaining placement and drawer-closing stages are omitted after the break mark.

stable execution. Perception can also proceed asynchronously during non-blocking periods, further concentrating overhead at meaningful decision boundaries.

The resulting trade-off is between the latency of these event-triggered coordination stages and the reliability gained from verifying subtask completion. In the illustrated rollout, verification is the dominant additional cost, whereas checkpoint recovery is comparatively short. EVA therefore spends additional computation selectively when the execution state requires it, rather than imposing verification and replanning overhead uniformly throughout every rollout.

Memory-QA. We additionally evaluate our memory module under the *Memory-QA* setting of RoboMME, where historical visual observations are accessed through QA-style memory retrieval and reasoning to derive action instructions, rather than directly specifying the action. This setting provides a complementary evaluation of whether the proposed memory representation can support reasoning over retrieved historical observations.

As shown in Table 9, our method achieves an average accuracy of 89.6% under Memory-QA, outperforming the strongest baseline, Gemini-2.5-Pro, by 41.7 percentage points. The corresponding category averages are 79.0%, 85.5%, 95.0%, and 99.0% for Counting, Permanence, Reference, and Imitation, respectively. It demonstrating its effectiveness on tasks that require reasoning about object identities, temporal correspondences, and demonstrated trajectories. These results indicate that the proposed memory formulation remains effective when memory is accessed indirectly through retrieval and question answering, rather than through explicit action-oriented subgoals.

Table 9: RoboMME accuracy (%) with the oracle planner.

Method	Counting				Permanence				Reference				Imitation				AVG
	Bin Fill	Pick Xtimes	Swing Xtimes	Stop Cube	Video Umsk	Button Umsk	Video UmskS	Button UmskS	Pick HighL	Video Repick	Video PlcBtn	Video PlcOrd	Move Cube	Insert Peg	Pattern Lock	Route Stick	
Gemini-2.5-Pro	78.00	80.00	58.00	**32.00**	86.00	82.00	18.00	12.00	78.00	28.00	50.00	60.00	66.00	32.00	4.00	2.00	47.88
GPT4o	50.00	58.00	32.00	18.00	**32.00**	60.00	28.00	34.00	30.00	14.00	18.00	20.00	70.00	18.00	6.00	2.00	30.63
Qwen3-VL-32B	54.00	60.00	54.00	**32.00**	26.00	28.00	22.00	14.00	14.00	32.00	26.00	14.00	66.00	2.00	2.00	0.00	27.88
EVA	**100.00**	**100.00**	**98.00**	18.00	**100.00**	**100.00**	**100.00**	**42.00**	**92.00**	**100.00**	**100.00**	**88.00**	**100.00**	**100.00**	**96.00**	**100.00**	89.60

E PROOFS

We provide proofs of the theoretical results stated in § 4. Throughout, let $r = p(1-q) + (1-p)q$ denote the probability that the monitor triggers a retry after a single execution. Since $p, q \in (0, 1)$, we have $0 < r < 1$.

Proposition 5.1 (Benefit of harness) With at most m execution attempts, the probability of successfully completing the task is $S_m = pq(1-r^m)/(1-r)$. As the number of retries approaches infinity, the success probability converges to $S_\infty = pq/(1-r)$. Moreover, the framework achieves a higher success probability than a standalone executor, i.e., $S_\infty > p$, if and only if $q > \frac{1}{2}$.

Proof. A task is successfully completed on a given attempt if and only if the underlying model execution succeeds and the monitor correctly identifies the execution as successful. Since the execution succeeds with probability p, and the monitor correctly identifies a successful execution with probability q, the probability of successful completion on any given attempt is pq. After an

unsuccessful or incorrectly monitored execution, a retry is triggered with probability $r = p(1-q) + (1-p)q$. Therefore, the probability that the system reaches the i-th attempt is r^{i-1}. The probability of successfully completing the task on the i-th attempt is consequently pqr^{i-1}. With at most m attempts, the overall success probability is therefore $S_m = \sum_{i=1}^{m} pq\, r^{i-1} = pq \sum_{i=0}^{m-1} r^i$.

Since $0 < r < 1$, the geometric-series formula gives

$$S_m = pq \frac{1 - r^m}{1 - r}.$$

Taking the limit as $m \to \infty$, we obtain

$$S_\infty = \lim_{m \to \infty} S_m = \frac{pq}{1 - r}.$$

Using the definition of r, $1 - r = 1 - p(1-q) - (1-p)q = 1 - p - q + 2pq$, and hence

$$S_\infty = \frac{pq}{1 - p - q + 2pq}.$$

It remains to determine when $S_\infty > p$. Since the denominator is positive,

$$\frac{pq}{1 - p - q + 2pq} > p,$$

which is equivalent to $(1-p)(2q-1) > 0$. Because $p \in (0,1)$, we have $1 - p > 0$. Thus,

$$S_\infty > p \quad \Longleftrightarrow \quad q > \frac{1}{2}.$$

This proves the proposition. □

Proposition 5.2 (Decay of retry gains) The marginal improvement for the m-th attempt is $\Delta S_m = S_m - S_{m-1} = pqr^{m-1}$. Since $0 < r < 1$, the marginal gains decay geometrically, with $\Delta S_{m+1} / \Delta S_m = r$.

Proof. From Proposition 4.1, $S_m = pq \frac{1-r^m}{1-r}$. Similarly, $S_{m-1} = pq \frac{1-r^{m-1}}{1-r}$. Therefore, the marginal improvement obtained by allowing the m-th attempt is

$$\Delta S_m = S_m - S_{m-1} = pq \frac{1 - r^m - (1 - r^{m-1})}{1 - r} = pqr^{m-1}.$$

Because $0 < r < 1$, the marginal gains form a geometric sequence. In particular,

$$\frac{\Delta S_{m+1}}{\Delta S_m} = \frac{pqr^m}{pqr^{m-1}} = r < 1.$$

Hence, each additional retry provides a strictly smaller improvement than the preceding retry, and the marginal gains decay geometrically. □

Proposition 5.3 (Execution-monitor trade-off) When $p + q > 1$, the marginal benefit of improving the component with higher accuracy is larger. Specifically,

$$\frac{\partial S_\infty}{\partial p} - \frac{\partial S_\infty}{\partial q} = \frac{(q-p)(1-p-q)}{(1-p-q+2pq)^2}.$$

Thus, if $p > q$, improving execution accuracy yields a larger marginal gain, while if $q > p$, improving verifier accuracy yields a larger marginal gain.

Proof. From Proposition 4.1, the unlimited-retry success probability can be written as $S_\infty = \frac{pq}{1-p-q+2pq}$. For convenience, define $D = 1 - p - q + 2pq$. Since $D = 1 - r > 0$, we may differentiate $S_\infty = pq/D$ with respect to p and q. First, with respect to p,

$$\frac{\partial S_\infty}{\partial p} = \frac{qD - pq(-1+2q)}{D^2}.$$

By simplify equation, $q(1-p-q+2pq) - pq(-1+2q) = q(1-q)$, and therefore

$$\frac{\partial S_\infty}{\partial p} = \frac{q(1-q)}{D^2}.$$

Similarly,

$$\frac{\partial S_\infty}{\partial q} = \frac{pD - pq(-1+2p)}{D^2} = \frac{p(1-p)}{D^2}.$$

Thus,

$$\frac{\partial S_\infty}{\partial p} - \frac{\partial S_\infty}{\partial q} = \frac{q(1-q) - p(1-p)}{D^2} = \frac{(q-p)(1-p-q)}{D^2}.$$

When $p + q > 1$, we have $1 - p - q < 0$. Since $D^2 > 0$, it follows that

$$\operatorname{sign}\left(\frac{\partial S_\infty}{\partial p} - \frac{\partial S_\infty}{\partial q}\right) = \operatorname{sign}(p-q).$$

Consequently, if $p > q$,

$$\frac{\partial S_\infty}{\partial p} > \frac{\partial S_\infty}{\partial q},$$

whereas if $q > p$,

$$\frac{\partial S_\infty}{\partial q} > \frac{\partial S_\infty}{\partial p}.$$

This proves the proposition. □

Proposition 5.4 (Performance under imperfect reset) When the reset operation succeeds with probability $k \in (0,1)$, Agent Harness improves over single-shot execution if and only if $q > (1-pk)/(1+k-2pk) > 1/2$. Thus, imperfect reset lowers the achievable performance and raises the monitoring accuracy required for a net gain.

Proof. Suppose that a retry is triggered with probability r, as before, but that the reset operation succeeds only with probability k. Thus, after a retry is triggered, the system reaches the next execution attempt with probability k. The probability of progressing from one attempt to the next is therefore kr. The probability of successfully completing the task on the first attempt remains pq. To succeed on the second attempt, the system must first trigger a retry and successfully reset, which occurs with probability rk, followed by a successful execution and correct monitoring, which occurs with probability pq. Hence, the probability of success on the i-th attempt is $pq(kr)^{i-1}$.

With unlimited retries, the total success probability is therefore

$$S_\infty^{(k)} = \sum_{i=1}^{\infty} pq(kr)^{i-1} = \frac{pq}{1-kr},$$

where $0 < kr < 1$. We now determine when imperfect-reset harnessing improves over single-shot execution. We require $S_\infty^{(k)} > p$, which is equivalent to $\frac{pq}{1-kr} > p$. Since $p > 0$, $q > 1 - kr$. Substituting $r = p(1-q) + (1-p)q$, we obtain $q > 1 - k\big[p(1-q) + (1-p)q\big]$.

Expanding the right-hand side gives $q > 1 - kp + kq - kqp - kq + kqp$, which, after collecting the terms involving q, yields $q(1+k-2pk) > 1 - pk$. Since $1 + k - 2pk > 0$, we obtain:

$$q > \frac{1-pk}{1+k-2pk}.$$

Finally, this threshold is strictly greater than $1/2$ whenever $k < 1$. Indeed,

$$\frac{1-pk}{1+k-2pk} - \frac{1}{2} = \frac{2(1-pk) - (1+k-2pk)}{2(1+k-2pk)} = \frac{1-k}{2(1+k-2pk)} > 0.$$

Therefore,

$$S_\infty^{(k)} > p \quad \Longleftrightarrow \quad q > \frac{1-pk}{1+k-2pk} > \frac{1}{2}.$$

Thus, imperfect reset reduces the probability of successfully reaching subsequent attempts and consequently increases the monitor accuracy required for the harness to provide a net benefit. □

F Deployments on Real-world.

We deploy EVA on the Kepler K2 humanoid robot using a three-node architecture consisting of a GPU computation server, an intermediate server, and the physical robot. This architecture separates high-level inference, robot communication, and hardware control. As illustrated in Fig 6, the GPU machine hosts the agent service, VLA/WAM policy servers, and task scheduler. Given a human instruction, the agent first decomposes the task into a sequence of subtasks, after which the scheduler routes each subtask to the corresponding VLA or WAM policy and queries the policy using the latest robot observations. The resulting action chunks are then streamed to the robot bridge running on the intermediate server.

The intermediate machine serves as the sole communication interface to the physical robot. It continuously receives proprioceptive states and visual observations from the robot, forwards the latest observations to the GPU machine, and executes the action chunks produced by the selected policy. In particular, the intermediate machine maintains a short action buffer and aligns newly received action chunks with the currently queued actions to avoid stale or overlapping commands. It then sends the resulting commands to the robot controller at a fixed control frequency. The physical robot is responsible only for publishing proprioceptive states and camera observations and receiving low-level control commands. This separation prevents the high-level inference stack from directly accessing the robot hardware, while the robot bridge provides a stable and hardware-specific execution interface.

For each subtask, the scheduler first brings the robot to the corresponding initial pose before requesting the first action chunk. During execution, the scheduler continuously collects multi-view observations, including an egocentric camera and wrist-mounted cameras, and feeds them to the selected VLA or WAM policy. The predicted action chunks are asynchronously streamed to the intermediate machine and executed by the robot. Upon subtask completion, failure, or termination, the scheduler reports the resulting progress observation to the world service, which is subsequently used by the agent for high-level task progression and recovery. This architecture decouples task-level reasoning and policy inference from hardware-level execution while enabling closed-loop interaction among the agent, the VLA/WAM policy, and the physical humanoid robot.

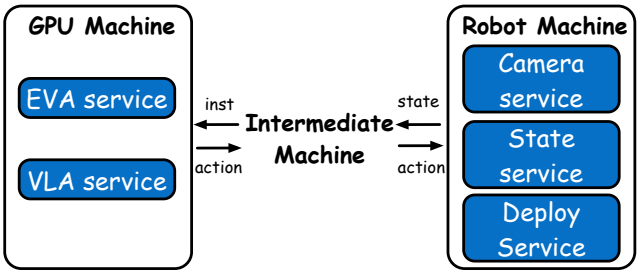

Figure 6: Web Interface for EVA

G Discussion

Beyond the Executor: A Design Philosophy. An important question is how EVA relates to the emerging paradigm of using foundation models such as GPT-6 as robot controllers, including

code-as-policy approaches. We argue that these paradigms are better viewed as complementary rather than competing alternatives. Code-based policies can leverage the reasoning and compositional capabilities of large language models to generate control programs, making them attractive for structured locomotion or manipulation. However, code generation and execution can introduce substantial inference and execution latency, which may be undesirable in real-world deployment. In contrast, VLA policies can directly map visual observations and language instructions to low-level actions, enabling fine-grained and reactive control. Nevertheless, their performance can degrade under distribution shifts, highlighting the gap between high-level semantic understanding and reliable physical execution.

This motivates a broader perspective: rather than seeking a universally superior executor, a robotic agent should coordinate heterogeneous executors according to their respective strengths and failure modes. A single task may naturally involve different control regimes, where a code-based controller is suitable for structured or temporally extended behaviors, while a VLA policy is preferable for dexterous and reactive manipulation. EVA provides an execution-level abstraction for such heterogeneous capabilities, allowing different executors to be seamlessly invoked within a unified framework. The Harness focuses on *how* task execution is managed—including task decomposition, execution monitoring, failure detection, recovery, and executor scheduling—rather than prescribing *what* executor should perform the task. Thus, instead of replacing increasingly capable robot foundation models, EVA serves as a common orchestration layer that composes complementary executors and can readily incorporate improvements from individual policies.

Future Directions. We believe that the next step toward more capable robotic agents is not simply to scale the underlying model, but to improve the entire execution loop. EVA provides a natural foundation for such an evolution, enabling three complementary directions.

First, the foundation model in harness of EVA can evolve through physical interaction. By connecting foundation models to the physical world through the Harness, models can move beyond passive reasoning and learn from actual execution outcomes. In particular, agentic reinforcement learning can use successful executions, failures, and recovery trajectories to improve a model's ability to decompose tasks, reason about physical consequences, select appropriate executors, and adapt its behavior based on feedback. In this way, the Harness provides a structured interface through which abstract reasoning can be grounded in physical experience.

Second, the Harness itself can evolve. Rather than treating the Harness as a fixed runtime, its monitoring, task decomposition, executor scheduling, failure diagnosis, and recovery strategies can also be learned and improved from accumulated execution experience. For example, repeated failures can reveal when an executor should be replaced, when additional monitoring is needed, or which recovery strategy is most effective. This turns the Harness from a static orchestration mechanism into an adaptive component of the robotic agent.

Third, improving the capabilities of low-level executors can substantially expand the capability range of VLA or WAM models. Much of the existing work on improving VLA or WAM focuses on enhancing the models themselves. However, the expressiveness of an agent is fundamentally constrained by what its underlying executor can physically accomplish. For example, an executor that can only perform primitive skills such as grasp a bottle" supports a very different range of tasks from one that can execute a more complex instruction such as prepare a full meal." Improving the capabilities of low-level executors can therefore directly broaden the space of behaviors available to the higher-level agent, enabling it to address increasingly complex and diverse tasks.

H Interface for Interaction

As shown in Figure 7, an interactive monitoring and debugging dashboard for an embodied agent harness, providing end-to-end visibility into the robot's perception, reasoning, planning, and execution pipeline. The interface integrates live robot observations, frame-level navigation, agent run history, VLM inputs and structured outputs, planner traces, visual descriptions, and persistent object memory into a unified workspace, enabling developers to inspect, validate, and debug the agent's behavior throughout real-world manipulation tasks.

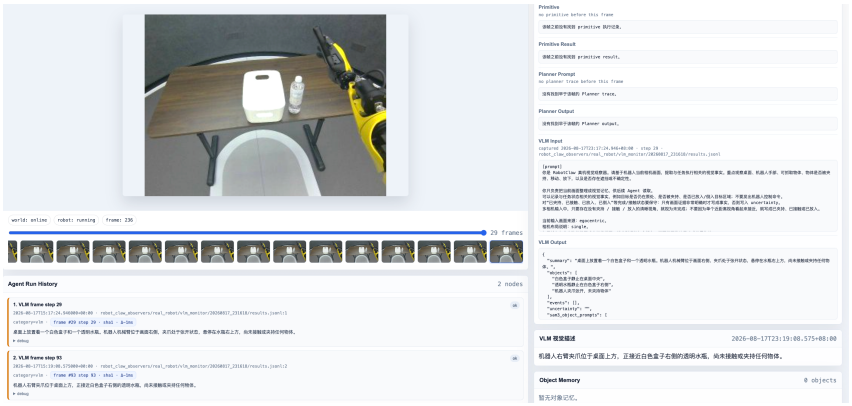

Figure 7: Web Interface for EVA