CO-HARNESS: COLLECTIVE HARNESS EVOLUTION FOR AGENTIC MULTIMODAL LARGE LANGUAGE MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing multimodal large language model (MLLM) agents generally rely on predefined contexts, fixed tool-use policies, and static orchestration to support long-horizon reasoning and tool use, but these rigid execution mechanisms limit their ability to adapt to new and diverse multimodal agentic tasks. In this work, we aim to improve MLLM agents through harness evolution, which abstracts generalizable experience from successful and failed execution trajectories to refine model harness for related but previously unseen tasks. To this end, we propose **Co-Harness**, a new framework that introduces collective learning into multimodal harness evolution to enhance the long-horizon reasoning and tool-use capabilities of multimodal agents. The core idea of Co-Harness is to jointly improve a set of specialized and complementary harnesses through shared experience and reciprocal feedback. Specifically, Co-Harness introduces two collective harness evolving methods: *Collective Mixture-of-Harness Evolution* (CoMixHE) and *Memory-guided Collective Harness Co-Evolution* (MemCoHE). CoMixHE collectively evolves specialized and complementary harnesses across and within task domains, expanding harness capacity and broadening the search space of harness optimization. MemCoHE jointly evolves the task model's harnesses and the optimization model's memory, enabling the optimizer to reuse past refinement experiences and adapt more effectively to changing failure patterns. Extensive experiments on multiple multimodal agentic benchmarks demonstrate the effectiveness of Co-Harness and its transferability across tasks and models.

1 INTRODUCTION

Multimodal large language models (MLLMs) jointly process visual and textual information to enable multimodal understanding and reasoning, showing considerable potential in robotics, personal assistants, and multimodal information retrieval (Bai et al., 2025; Qwen Team, 2026; Zhang et al., 2025b; Yao et al., 2024; Meng et al., 2025). Recent research has increasingly moved beyond conventional tasks, such as static visual question answering (Antol et al., 2015) and visual groundingYu et al. (2016), toward more complex agentic tasks involving long-horizon planning, tool use, code generation, and web search (Guo et al., 2025; Li et al., 2025a; Tao et al., 2026; Su et al., 2026) as in Fig. 1 (a). Despite this progress, existing MLLMs still rely heavily on: (1) predefined contexts that provide natural language guidance for task execution; (2) fixed tool-use policies that govern tool selection and invocation; and (3) static agent loop that coordinate model calls throughout the reasoning process. Together, these components constitute an MLLM's *harness*. Such rigid designs limit MLLM's ability to flexibly adapt tool use and sustain effective long-horizon reasoning when confronted with new and diverse multimodal agentic tasks (Jiang et al., 2026).

Recent advances in large language models (LLMs) have demonstrated the potential of leveraging past agent experiences to evolve an agent's context, tool-use policies, and agent loop, thereby improving its capabilities in long-horizon reasoning and tool use (Lee et al., 2026; Novikov et al., 2025; Zhang et al., 2026). A central idea underlying these advances is "Harness Evolution", which first extracts generalizable behavioral patterns from successful and failed execution trajectories: which strategies work, under what conditions they should be applied, and which mistakes should be avoided. The extracted knowledge is then used to update the agent's context, tool-use policies,

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.

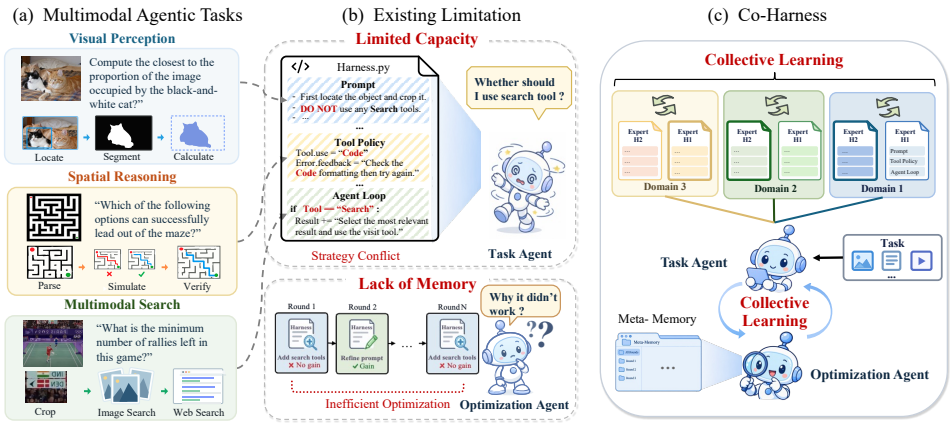


Figure 1: **Motivation.** (a) Multimodal agentic tasks require heterogeneous capabilities, such as visual perception, spatial reasoning and multimodal search. (b) Existing harness evolution methods face two key challenges: *Limited Harness Capacity*, as a single harness struggles to accommodate diverse and potentially conflicting task-solving strategies, and *Lack of Harness Optimization Memory*, which leads to repeated ineffective optimization attempts. (c) Co-Harness collectively evolves specialized and complementary expert harnesses within and across task domains, while accumulating refinement experience in a meta-memory to enable co-evolution of task and optimization agents.

and agent loop, enabling these evolved components to guide the LLM when solving related but previously unseen tasks.

An intuitive idea is to directly apply these LLM harness evolution methods to continuously improve multimodal agents, which, however, does not perform well. As illustrated in Fig. 1 (b), we attribute this limitation primarily to two challenges. (1) **Limited Harness Capacity**: Most existing LLM harness evolution methods typically maintain a single harness that continually integrates various strategies learned from diverse tasks (Lee et al., 2026; Zhang et al., 2026). Multimodal agentic tasks, however, require heterogeneous capabilities, such as visual perception, spatial understanding, cross-modal reasoning, and multimodal tool interaction. A single harness may thus struggle to accommodate diverse and potentially conflicting multimodal strategies, causing task interference and limiting performance. (2) **Lack of Harness Optimization Memory**: Most existing methods (Lee et al., 2026; Karten et al., 2026) typically use an optimization agent to refine the task agent's harness without explicitly retaining experience from previous refinement attempts. Without a continually updated memory of optimization experience, the optimization agent relies primarily on the knowledge encoded in its fixed parameters. This limits its ability to recognize new failure patterns, reuse previously effective refinement strategies, and avoid repeating unsuccessful optimization attempts. This issue is particularly challenging for multimodal agents operating across diverse tasks and changing physical environments, limiting the adaptability and effectiveness of harness evolution.

To tackle these challenges, we propose **Co-Harness**, a new framework that introduces collective learning into multimodal harness evolution to enhance the long-horizon reasoning and tool-use capabilities of multimodal agents as in Fig. 1 (c). The core idea of Co-Harness is to jointly improve a set of specialized and complementary harnesses through shared experience and reciprocal feedback. Specifically, Co-Harness designs a new harness structure, i.e., a mixture of harnesses, which maintains and continuously updates a set of harnesses and dynamically routes each task to an appropriate harness or composes multiple harnesses as needed during inference. Upon this structure, Co-Harness introduces two types of collective harness evolution methods, including Collective Mixture-of-Harness Evolution (CoMixHE) and Memory-Guided Collective Harness Co-Evolution (MemCoHE).

CoMixHE works by enabling expert harnesses to learn from one another's execution experience and complementary feedback. It operates at two levels: (a) Cross-domain CoMixHE that maintains different harnesses for tasks from different domains, each with capabilities and strategies specific to its domain; and (b) Within-domain CoMixHE that models each domain harness as a set of expert

harnesses, where a novel Expert Harness Discrimination Loss have been introduced to encourage experts to explore, identify and develop complementary problem-solving capabilities and strategies within the same domain. With this two-level collective learning process, CoMixHE maintains multiple harnesses and encourages them to generate diverse execution trajectories and share complementary experiences across and within domains, enabling each harness to evolve by learning from others' successes and failures, ultimately expanding harness capacity and broadening the search space of harness optimization.

MemCoHE introduces a bidirectional evolution mechanism between the task agent's harnesses and the optimization agent's memory, where the optimization agent refines the task agent's harnesses according to task execution outcomes, while the refinement results of task agent's harness (e.g., improvements and regressions) are used to update the optimization agent's meta memory of harness optimization strategies. This bidirectional process enables the optimization agent to accumulate experience, recognize new failure patterns, reuse previously effective refinement strategies, and avoid repeating unsuccessful attempts, thereby improving the adaptability and efficiency of harness evolution across diverse multimodal tasks and changing physical environments.

The main contributions of this work are fourfold. First, we introduce Co-Harness, which, to the best of our knowledge, is the first work that explores collective learning for multimodal harness evolution, enabling multimodal agents to continually improve through shared interaction experience. Second, we propose CoMixHE, which organizes complementary expert harnesses across and within domains, expanding harness capacity and broadening the search space of harness optimization. Third, we propose MemCoHE, which jointly evolves the task agent's harnesses and the optimization agent's memory, improving the optimization agent's adaptability by leveraging past refinement experiences. Fourth, extensive experiments on multiple multimodal agentic benchmarks demonstrate the effectiveness of Co-Harness and its transferability across tasks and models.

2 Related Work

2.1 Multimodal Agentic Reasoning

Research on MLLMs is increasingly shifting from static, single-turn visual question answering (Antol et al., 2015) toward more complex agentic tasks , where agents are required to perform multi-step reasoning (Su et al., 2025; Guo et al., 2025), write code for visual manipulation (Zheng et al., 2025; Li et al., 2025a), and conduct web search (Yao et al., 2026; Su et al., 2026). In this process, agent behavior is determined not only by the MLLM, but also by the external agent harness, including context organization, tool-use policies, and orchestration (Lee et al., 2026; Su et al., 2026). However, most multimodal agents remain fixed across episodes, limiting their ability to learn from prior experience and avoid repeated trial and error (Jiang et al., 2026). Recent works accumulate reusable experience through external skills or memories (Jiang et al., 2026; Xiong et al., 2026). However, they mainly adapt the context while keeping the harness fixed, still relying on the model to interpret and execute retrieved experience. In this work, we explore multimodal harness optimization, enabling agents to leverage execution trajectories and feedback to continually evolve their context organization, tool-use policies, and execution workflows, thereby enhancing the cross-task adaptability of multimodal agents.

2.2 Harness Engineering

Agent harnesses serve as the operational layer that turns foundation models into closed-loop agents (Lee et al., 2026; Martin, 2026). They determine what information from prior interactions is retained, how intermediate directives are formed, which context are exposed at each stage, and how the overall execution process proceeds. Recent studies (Lee et al., 2026; Tian et al., 2026) further show that, even with the underlying LLM fixed, changing the surrounding harness can lead to performance differences of up to $6\times$ on the same benchmark, highlighting the critical role of harness design in agent performance. Motivated by this observation, a growing body of work has explored automatic harness evolution. Existing studies span coding agents (Zhang et al., 2025a; Liu et al., 2026) as well as text reasoning and general-purpose agents (Lee et al., 2026; Ye et al., 2026; Zhang et al., 2026), where execution feedback is used to iteratively refine prompts, memory, tool-use policies or workflows. However, harness evolution for complex multimodal agentic tasks remains

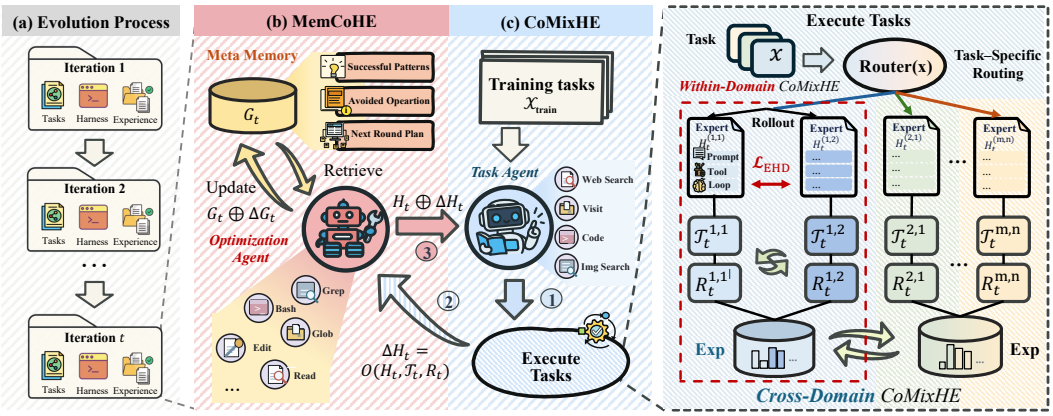


Figure 2: **Overview of Co-Harness.** **(a)** Co-Harness designs a new mixture of harnesses, which maintains and continuously updates a set of harnesses and dynamically routes each task to an appropriate harness or composes multiple harnesses as needed during inference. It evolves via MemCoHE and CoMixHE. **(b)** MemCoHE introduces a bidirectional evolution mechanism, which refines task agent's harnesses with task execution outcomes while the refinement results, in turn, are used to update the optimization agent's meta memory of harness optimization strategies. **(c)** CoMixHE works by enabling expert harnesses to learn from one another's execution experience and complementary feedback across and within domains.

underexplored. Such agents must jointly manage visual context, code execution, visual tools, and web interaction (Guo et al., 2025; Li et al., 2025a; Su et al., 2026), resulting in a substantially more complex design space and execution dependency structure. Systematically leveraging cross-task execution experience to continually evolve multimodal agent harnesses therefore remains an open problem.

3 METHODOLOGY

This section presents Co-Harness, a new framework that introduces collective learning into multimodal harness evolution as in Fig. 2, which enables a group of harnesses to co-evolve through shared execution experience and accumulated optimization feedback. In the following subsections, we first define the multimodal agent harness, formulate the evolution objective and describe the iterative evolution loop in which the task agent executes tasks and the optimization agent refines the task agent's harnesses based on execution feedback. We then introduce two collective harness evolution methods: Collective Mixture-of-Harness Evolution (CoMixHE), which enables specialized and complementary task harnesses to learn from shared experience, and Memory-Guided Collective Harness Co-Evolution (MemCoHE), which jointly evolves the task agent's harnesses and the optimization agent's meta-memory based on refinement outcomes.

Multimodal Agent Harness. Following Anthropic's characterization (Martin, 2026), we define a multimodal agent harness as a stateful program that wraps a multimodal large language model (MLLM) and manages its multimodal context, tool use, and agent loop during task execution. Pairing an MLLM M with a harness H yields a multimodal task agent that can invoke tools to execute tasks.

3.1 CO-HARNESSS

Evolution Objective. We represent each multimodal agentic task as $\mathcal{X} = (I, T)$, where I is an image and T is a textual instruction. Given a frozen MLLM M, our goal is to evolve the harness H to maximize the model's performance on the target tasks. For a harness H and s task instance $x \sim \mathcal{X}$, we execute a rollout trajectory $\tau \sim (M, H, x)$. During the rollout, the harness constructs prompts for M, receives the model's responses, and updates its state after each interaction. A task-specific reward function $R(\tau, x)$ scores the resulting trajectory, while the objective of harness optimization

is to find the harness that maximizes the expected final reward:

$$H^* = \arg\max_{H} \mathbb{E}_{x\sim\mathcal{X},\,\tau\sim(M,H,x)}\, R(\tau, x),$$

Evolution Loop. Co-Harness formulates harness evolution for target multimodal agentic tasks as a feedback-driven iterative optimization process (Lee et al., 2026; Novikov et al., 2025; Zhang et al., 2026). Given a fixed MLLM M and a training task set $\mathcal{X}_{train}$, this process alternates between task Specifically, at iteration t, the task agent combines M with a multimodal task-specific harness Su et al. (2026) H_t supporting search, webpage visiting, and code execution, to execute tasks and collect a trajectory set $\mathcal{T}_t$ and environmental feedback R_t. The optimization agent O is instantiated by a MLLM with file-management harness Google (2025) for file inspection and code editing, with tools such as `bash`, `grep`, and `glob`. It analyzes these trajectories and feedback to identify successful patterns and strategies for avoiding failures, then generates actionable textual gradients (Yuksekgonul et al., 2025) to refine task agent's harness. The resulting modifications cover task prompts, tool-policy and agent loop. We denote these harness modifications by

$$\Delta H_t = O(H_t, \mathcal{T}_t, R_t). \tag{1}$$

The harness is then updated as

$$H_{t+1} = H_t \oplus \Delta H_t, \tag{2}$$

where $\oplus$ denotes applying the proposed modifications to the current task agent's harness. By editing the task agent's harness code, the optimization agent incorporates these strategies into H_{t+1} to improve task agent execution in the next iteration. The resulting feedback guides further harness optimization, supporting progressive improvements in the agent's task-solving capabilities across multiple rounds. For each evolution harness, we retain its checkpoint from every iteration and select the checkpoint with the best training-set performance for test-time execution.

3.2 Collective Mixture-of-Harness Evolution

Prior methods (Lee et al., 2026; Zhang et al., 2026) optimize a single shared harness over a target task distribution, risking strategy interference and conflicting updates across heterogeneous multimodal agentic tasks. We therefore propose Collective Mixture-of-Harness Evolution (CoMixHE). Inspired by Mixture-of-Experts (MoE) (Jacobs et al., 1991; Fedus et al., 2022; Wang et al., 2025a), we introduce a Mixture of Harnesses (MoH) that maintains a set of specialized and complementary harnesses and enables them to evolve collectively through shared execution experience as shown in Fig. 2 (c). CoMixHE operates at two levels: Cross-domain CoMixHE, which enables experience sharing among domain-specific harnesses, and Within-domain CoMixHE develops complementary expert harnesses within each domain, enabling different experts to learn from one another's complementary execution trajectories.

Cross-domain CoMixHE. Specifically, Cross-domain CoMixHE initializes k domain-specific harnesses, $\{H_0^{(d)}\}_{d=1}^{k}$, all of which share the same base initialization. At each evolution iteration t, a router agent, also initialized from the task agent, selects a domain $d = \mathrm{Router}(x)$ for each task x based on its type and assigns the task to the corresponding harness $H_t^{(d)}$. The task agent executes the assigned tasks, producing a trajectory set $\mathcal{T}_t^{(d)}$ and corresponding environmental feedback $R_t^{(d)}$. The optimization agent jointly analyzes each harness's own execution trajectories $\mathcal{T}_t^{(\neg d)}$ and feedback $R_t^{(\neg d)}$ shared by the harnesses from all other domains. The textual update gradient can be formulated as follows:

$$\Delta H_t^{(d)} = O\left(H_t^{(d)}, \underbrace{\mathcal{T}_t^{(d)}, R_t^{(d)}}_{\text{Domain-specific}}, \underbrace{\mathcal{T}_t^{(\neg d)}, R_t^{(\neg d)}}_{\text{Cross-domain}}\right). \tag{3}$$

This collective experience provides the optimization agent with richer cross-domain feedback signals, enabling each harness to learn a broader range of generalizable success and failure patterns from other domains.

Within-domain CoMixHE. Within-domain CoMixHE further decomposes each domain-specific harness $H_t^{(d)}$ into N expert harnesses, denoted by $\{H_t^{(d,i)}\}_{i=1}^{N}$. Within each domain, experts execute tasks from different perspectives, and each expert learns from various with-domain execution

trajectories $\mathcal{T}_t^{(d,\neg i)}$ and feedback $R_t^{(d,\neg i)}$ shared by the other experts. In addition, to encourage complementarity among experts within the same domain, we propose a novel Expert Harness Discrimination (EHD) Loss, a textual loss that assesses differences among expert harnesses within the same domain:

$$\mathcal{L}_{\text{EHD}}\left(H_t^{(d)}\right) = \mathcal{J}\left(\left\{\left(H_t^{(d,i)}, \mathcal{T}_t^{(d,i)}\right)\right\}_{i=1}^{N}\right), \tag{4}$$

where $\mathcal{J}$ denotes the optimization agent's assessment of differences in the experts' problem-solving patterns based on their harness code and execution trajectories. To avoid additional computational overhead, we carefully prompt the optimization agent to implicitly assess these differences by inspecting the experts' harness code and execution trajectories, incorporating the resulting feedback directly into the textual optimization gradients. In CoMixHE, the optimization agent combines the resulting textual feedback with execution experience to jointly refine the task agent's expert harnesses and encourage complementary strategies, as follows:

$$\Delta H_t^{(d,i)} = O\Bigg(H_t^{(d,i)}, \underbrace{\mathcal{T}_t^{(d,i)}, R_t^{(d,i)}}_{\text{Own}}, \underbrace{\mathcal{T}_t^{(d,\neg i)}, R_t^{(d,\neg i)}}_{\text{Within-Domain}}, \underbrace{\mathcal{T}_t^{(\neg d)}, R_t^{(\neg d)}}_{\text{Cross-Domain}}, \mathcal{L}_{\text{EHD}}\Bigg), \tag{5}$$

In this way, CoMixHE increases collective harness capacity through domain specialization and expert complementarity. By generating diverse execution trajectories and sharing experience across and within domains, each harness learns from others' successes and failures to guide its own evolution. This collective learning exposes the optimization agent to a broader range of strategies and refinement directions, expanding the search space for harness optimization.

3.3 Memory-guided Collective Harness Co-Evolution

Without accumulated optimization memory, the optimization model relies primarily on the fixed knowledge encoded in its parameters, limiting its ability to identify new failure patterns, reuse effective refinement strategies, and avoid repeated optimization failures. We therefore propose Memory-guided Collective Harness Co-Evolution (MemCoHE) As shown in Fig. 2(b), which jointly evolves task harnesses and optimization memory: the optimizer refines harnesses using execution feedback and accumulated memory, while refinement outcomes update this memory to guide subsequent optimization. Specifically, MemCoHE maintains a persistent optimization meta-memory G_t as a collection of files, rather than solely as textual input to the optimization agent. These files contain past successful patterns, avoided operations, and next round explore plan. The meta-memory evolves through incremental updates:

$$G_{t+1} = G_t \oplus \Delta G_t, \tag{6}$$

where ΔG_t denotes the proposed meta-memory optimization. The optimization agent generates these modifications by reviewing past harnesses and their updates alongside the accumulated meta-memory:

$$\Delta G_t = O\left(G_t, \{(H_j, \Delta H_j)\}_{j=0}^{t-1}\right). \tag{7}$$

Here, H_j and ΔH_j denote the harness and its modifications at iteration j, respectively. The updated meta-memory consolidates past refinement experience to guide subsequent harness optimization. When generating harness optimization, the optimization agent selectively reads and refines the meta-memory files in G_t within its coding workspace, drawing on current execution trajectories and feedback. The harness refinement history, in turn, guides subsequent meta-memory updates, forming an iterative loop between harness evolution and meta-memory evolution.

4 Experiments

4.1 Experiments Details

Implementation Details. We use Gemini-3-Flash (Doshi, 2025), Gemini-2.5-Pro (Kavukcuoglu, 2025), and GPT-5-mini (OpenAI, 2025) as base MLLM. The Gemini models evolve harnesses using their own trajectories, while GPT-5-mini uses Gemini-3-Flash's experience to assess cross-model generalization. The initial task-agent harness follows XSkill (Jiang et al., 2026), providing access to Web Search, Image Search, Visit, and Code tools. The optimization agent uses Gemini CLI Google

Method	VisualToolBench		TIR-Bench		MMSearch-Plus		AgentVista		Average	
	Avg.@4	Pass@4	Avg.@4	Pass@4	Avg.@4	Pass@4	Avg.@4	Pass@4	Avg.@4	Pass@4
Gemini-3-Flash										
CoT	25.12	36.92	28.50	54.00	16.47	24.64	18.35	29.36	22.11	36.23
Tools	41.94	60.75	32.37	58.50	39.57	53.55	20.64	39.45	33.63	53.06
AWM	41.94	59.35	34.25	62.50	43.36	54.98	19.50	37.61	34.76	53.61
DC	41.70	59.81	33.75	59.00	40.28	55.45	20.18	36.70	33.98	52.74
Agent-KB	41.75	61.21	36.62	62.00	39.81	53.08	21.33	38.53	34.88	53.71
XSkill	46.50	64.02	47.75	75.00	43.72	56.40	23.39	40.37	40.34	58.95
Meta-Harness	50.23	67.29	49.13	77.50	46.92	58.29	26.61	44.04	43.22	61.78
Co-Harness	**53.62**	**72.43**	**54.25**	**79.50**	**51.07**	**62.09**	**31.19**	**46.79**	**47.53**	**65.20**
Gemini-2.5-Pro										
CoT	20.91	28.97	21.37	40.00	10.43	19.43	17.89	28.44	17.65	29.21
Tools	25.35	40.65	28.38	54.00	21.56	35.55	20.18	33.94	23.87	41.04
AWM	25.93	39.25	29.75	53.50	20.85	36.97	19.72	32.11	24.06	40.46
DC	24.77	37.38	27.62	51.00	24.64	40.76	21.79	35.78	24.71	41.23
Agent-KB	26.75	41.12	29.13	52.50	23.22	37.91	20.87	33.94	24.99	41.37
XSkill	30.49	46.73	33.12	58.00	27.96	44.08	22.94	34.86	28.63	45.92
Meta-Harness	31.07	48.13	31.13	59.00	28.08	48.34	25.23	40.37	28.88	48.96
Co-Harness	**35.51**	**50.47**	**36.38**	**64.00**	**32.23**	**50.71**	**27.52**	**43.12**	**32.91**	**52.08**
GPT-5-mini										
CoT	13.90	22.90	20.00	46.50	3.08	6.64	18.58	28.44	13.89	26.12
Tools	24.30	37.85	23.50	50.50	14.22	20.38	20.41	35.78	20.61	36.13
AWM	23.25	34.58	24.25	53.00	14.81	20.85	19.04	34.86	20.34	35.82
DC	23.83	35.05	24.50	53.50	13.74	18.48	21.10	36.70	20.79	35.93
Agent-KB	24.77	38.32	25.13	53.00	13.63	19.91	19.95	34.86	20.87	36.52
XSkill	24.53	37.85	28.25	56.00	16.11	23.22	23.85	38.53	23.19	38.90
Meta-Harness	25.47	39.25	29.13	58.00	15.52	22.27	25.46	39.45	23.90	39.74
Co-Harness	**30.49**	**47.66**	**31.63**	**61.00**	**20.97**	**27.01**	**28.44**	**44.04**	**27.88**	**44.93**

Table 1: Comparison with agent evolution methods across four multimodal agentic benchmarks. Each block is labeled by the task agent's model. Bold indicates the best score in each column within each block.

(2025) as its harness to inspect code and invoke tools. Harness and tool details are provided in Appendix A.1.

For both evolution and inference, we use temperature 1.0, top-p 1.0, and limits of 20 interaction turns, 8,192 tokens per turn, and 100 input images. We run eight evolution rounds and select the harness with the best training-set performance for testing. By default, we initialize three domains with two experts each. Each candidate harness is evaluated with one rollout per training sample, while testing uses four independent rollouts per sample. Further details are provided in Appendix C.1.

Datasets and Benchmarks. We evaluate on five multimodal agentic benchmarks. *VisualToolBench* (Guo et al., 2025) emphasizes multi-turn visual tool use, while *TIR-Bench* (Li et al., 2025a) focuses on code-based image processing. *MMSearch-Plus* (Tao et al., 2026) requires information integration and multimodal search, and *AgentVista* (Su et al., 2026) covers these task types. We additionally use *MMBrowseComp* (Li et al., 2025b) to assess generalization to out-of-distribution multimodal search tasks. We follow XSkill (Jiang et al., 2026) for training/test splits and tool configurations; details are provided in Appendix A.2.

Baselines. We consider two baseline settings: Reasoning with CoT (OpenAI, 2024b) and Tool-Integrated Reasoning Su et al. (2026). Building on Tool-Integrated Reasoning, we further compare Co-Harness with five agent evolution baselines. These include four context-level evolution methods: Agent Workflow Memory (AWM) (Wang et al., 2025b), Dynamic Cheatsheet (DC) (Suzgun et al., 2026), Agent-KB (Tang et al., 2025), and XSkill (Jiang et al., 2026), as well as the harness-level evolution method Meta-Harness (Lee et al., 2026). For a fair comparison, all evolution methods use the same training sets and task-level inference settings. Implementation details are provided in Appendix B.1.

Metrics. Following XSkill (Jiang et al., 2026), we report Average@4, the mean success rate across four independent rollouts reflecting performance stability, and Pass@4, the fraction of tasks solved in at least one rollout indicating the empirical capability upper bound. We use GPT-4o (OpenAI, 2024a) to judge final answers with binary scores (0 or 1).

4.2 Main Results

Main Comparison. As shown in Tab. 1, Tool-Integrated Reasoning consistently improves on CoT Reasoning by a large margin, underscoring the need for external tools in these multi-step multimodal tasks. Building on the Tools baseline, Co-Harness leverage its mixture-of-harness design and two-level collective learning, Co-Harness improves Avg@4/Pass@4 over baseline by +13.90%/+12.14% on Gemini-3-Flash and +9.04%/+11.04% on Gemini-2.5-Pro. Context-level evolving methods leave the harness fixed, so their gains depend on how well the model uses retrieved guidance. Accordingly, AWM (Wang et al., 2025b), DC (Suzgun et al., 2026), and Agent-KB (Tang et al., 2025) yield only limited gains on Gemini-3-Flash and Gemini-2.5-Pro, with no Avg@4 or Pass@4 increase beyond +2.5%. Even XSkill (Jiang et al., 2026), with carefully designed experience and skill retrieval and an additional embedding model, still below Co-Harness by 7.19%/6.25% in Avg@4/Pass@4 on Gemini-3-Flash and by 4.28%/6.16% on Gemini-2.5-Pro. Meta-Harness (Lee et al., 2026) improves by optimizing the full harness space. However, its single-harness design lead to capability conflicts and restrict the optimization space, leaving its Avg@4/Pass@4 scores 4.31%/3.42% below Co-Harness on Gemini-3-Flash. Without explicit meta-memory, Meta-Harness may struggle to generate effective harness edits, on Gemini-2.5-Pro, Meta-Harness shows only limited gains over XSkill, with Avg@4/Pass@4 scores 4.03%/3.12% below Co-Harness.

Transfer Comparison. To assess evolution robustness, we transfer the contexts or harnesses evolved with Gemini-3-Flash to GPT-5-mini without target-model evolution. As shown in Tab. 1, Co-Harness improves Avg@4/Pass@4 over Tools Baseline by +7.27%/+8.80%, suggesting that its learned execution strategies generalize beyond the source model. Context-level methods may transfer less effectively because their gains depend on how the target model interprets retrieved guidance; Only XSkill improves performance by +2.58%/+2.77%. Meta-Harness gains only +3.29%/+3.61%, because a single harness limits strategy diversity and favors source-model-specific execution patterns.

Method	Avg@4	Pass@4
Baseline	41.94	60.75
CoMixHE		
+ Mixture of Harness	50.58$^{+8.64}$	69.16$^{+8.41}$
+ Cross Domain CoMixHE	51.05$^{+9.11}$	70.09$^{+9.34}$
+ Within Domain CoMixHE	52.10$^{+10.16}$	71.50$^{+10.75}$
MemCoHE		
+ Meta Memory	53.62$^{+11.68}$	72.43$^{+11.68}$

Table 2: Ablation study on VisualToolBench using Gemini-3-Flash. Green superscripts show percentage-point gains over the baseline.

Domain	Expert	Avg@4	Pass@4
3	**2**	**53.62**	**72.43**
1	2	51.52$^{-2.10}$	69.16$^{-3.27}$
6	2	53.04$^{-0.58}$	71.96$^{-0.47}$
3	1	50.12$^{-3.50}$	67.76$^{-4.67}$
3	6	53.39$^{-0.23}$	72.90$^{+0.47}$

Table 3: Effect of initial domain and expert counts on VisualToolBench with Gemini3-flash. Red/green marks decreases/increases from the default setting (gray line).

4.3 Ablation Studies

We conduct an ablation study to assess the contribution of each component as illustrated in Tab. 2. Mixture-of-harness evolution improves Avg@4/Pass@4 by +8.64%/+8.41% over Tools, supporting the effectiveness of the Co-Harness architecture. Cross-domain CoMixHE enables agents to collectively learn successful patterns shared across domains, raising these gains to +9.11%/+9.34%. Within-domain CoMixHE proposes multiple experts per domain, diversifying trajectories and feedback and broadening the search space, resulting in gains of +10.16%/+10.75%. Finally, MemCoHE uses accumulated meta-memory to guide the optimizer toward more effective refinement strategies, further increasing the gains to +11.68%/+11.68%.

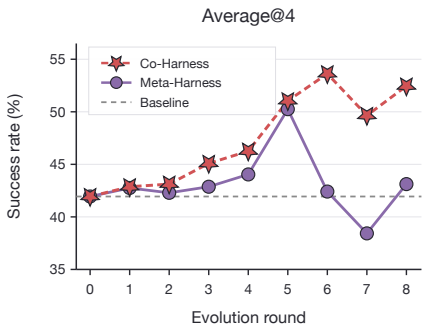


Figure 3: Average@4 performance across evolution rounds on VisualToolBench with Gemini-3-Flash.

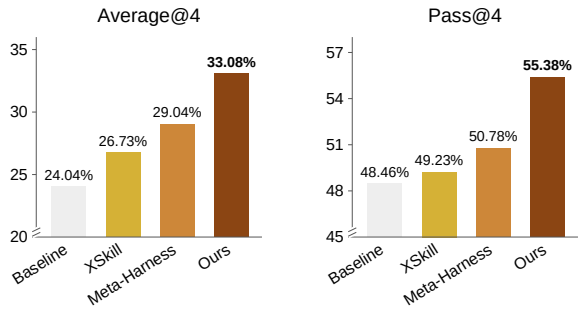


Figure 4: Cross-benchmark transfer from MMSearch-Plus to MMBrowseComp. Avg@4 (left) and Pass@4 (right) are reported for our and other baseline.

4.4 Discussion

Hyperparameter Analysis. We analyze the impact of hyperparameter settings on harness evolution by varying the initial harness capacity. As shown in Tab. 3, although the optimizer can add domains and experts during evolution, starting with single domain or single expert reduces Avg@4/Pass@4 by 2.10%/3.27% and 3.50%/4.67%, respectively, suggesting that limited initial capacity constrains early exploration. Increasing the domain count to six yields changes of -0.58%/-0.47%, while using six experts per domain yields -0.23%/+0.47%, showing no consistent benefit. Performance is relatively insensitive to larger domain and expert counts, supporting our default of three domains with two experts each to avoid extra optimization time at initialization.

Harness Evolution Dynamics. To assess whether the optimization agent benefits from prior exploration, we evaluate the candidate harness after each evolution round on VisualToolBench. As shown in Fig. 3, Meta-Harness reaches a peak Avg@4 of 50.23% at round 5 but fails to sustain this gain. Its performance drops to 38.43% at round 7, below the baseline. This regression suggests that access to the previous workspace alone may be insufficient to guide subsequent updates without explicit experience consolidation and refinement planning. By contrast, guided by its continually updated meta-memory, Co-Harness follows an overall upward trajectory, reaching 53.62% Avg@4 at round 6 while limiting unproductive exploration. This contrast suggests that MemCoHE helps the optimizer benefit from past exploration.

Out-of-Distribution (OOD) Task Evaluation. To evaluate generalization, we transfer harnesses evolved on MMSearch-Plus to Out-of-Distribution tasks MMBrowseComp evaluation. As shown in Fig. 4, Co-Harness achieves 33.08%/55.38% in Avg@4/Pass@4, exceeding XSkill by 6.35%/6.15% and Meta-Harness by 4.04%/4.60%. These results suggest that MoH and collective learning support stronger OOD generalization, whereas XSkill's context-level guidance and Meta-Harness's single-harness capacity limit transfer.

5 Conclusion

This paper presents **Co-Harness**, a new framework that introduces collective learning into multimodal harness evolution to enhance the long-horizon reasoning and tool-use capabilities of multimodal agents. Specifically, Co-Harness introduces two collective harness evolution methods, i.e., Collective Mixture-of-Harness Evolution (CoMixHE), which develops specialized and complementary expert harnesses across and within task domains through shared execution experience, and Memory-Guided Collective Harness Co-Evolution (MemCoHE), which accumulates past refinement experience in a persistent meta-memory to guide subsequent harness updates. In this way, Co-Harness mitigates the limited capacity of a single shared harness and the lack of harness optimization memory, enabling the task and optimization agents to co-evolve without updating the underlying model parameters. Extensive experiments across multiple multimodal agentic benchmarks and model families demonstrate the superiority of the proposed Co-Harness.

AI USE STATEMENT

We used generative AI tools to assist with language editing and improve the manuscript's clarity and readability. The authors reviewed and revised all AI-assisted content and take full responsibility for the accuracy and integrity of the final manuscript.

REPRODUCIBILITY STATEMENT

We detail the agent harnesses, tool configurations, and initial system prompts used by Co-Harness in Appendices A.1 and A.3. Benchmark configurations and evaluation protocols are provided in Appendices A.2 and B.2. Harness evolution settings and baseline implementations are described in Appendices C.1 and B.1. Harness checkpoints are selected based on training-set performance and subsequently evaluated on the test sets.

REFERENCES

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015. URL https://openaccess.thecvf.com/content_iccv_2015/html/Antol_VQA_Visual_Question_ICCV_2015_paper.html.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Tulsee Doshi. Gemini 3 Flash: Frontier intelligence built for speed. Google Blog, 2025. URL https://blog.google/products-and-platforms/products/gemini/gemini-3-flash/.

William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. URL https://www.jmlr.org/papers/v23/21-0998.html.

Google. Gemini CLI: An open-source ai agent. Software and documentation, 2025. URL https://github.com/google-gemini/gemini-cli.

Xingang Guo, Utkarsh Tyagi, Advait Gosai, Paula Vergara, Jayeon Park, Ernesto Gabriel Hernández Montoya, Chen Bo Calvin Zhang, Bin Hu, Yunzhong He, Bing Liu, and Rakshith Sharma Srinivasa. Beyond seeing: Evaluating multimodal LLMs on tool-enabled image perception, transformation, and reasoning, 2025. URL https://arxiv.org/abs/2510.12712.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79. URL https://doi.org/10.1162/neco.1991.3.1.79.

Guanyu Jiang, Zhaochen Su, Xiaoye Qu, and Yi R. Fung. XSkill: Continual learning from experience and skills in multimodal agents, 2026. URL https://arxiv.org/abs/2603.12056.

Seth Karten, Joel Zhang, Tersoo Upaa, Jr., Ruirong Feng, Wenzhe Li, Chengshuai Shi, Chi Jin, and Kiran Vodrahalli. Continual harness: Online adaptation for self-improving foundation agents, 2026. URL https://arxiv.org/abs/2605.09998.

Koray Kavukcuoglu. Gemini 2.5: Our most intelligent AI model. Google Blog, 2025. URL https://blog.google/innovation-and-ai/models-and-research/google-deepmind/gemini-model-thinking-updates-march-2025/.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-Harness: End-to-end optimization of model harnesses, 2026. URL https://arxiv.org/abs/2603.28052.

Ming Li, Jike Zhong, Shitian Zhao, Haoquan Zhang, Shaoheng Lin, Yuxiang Lai, Chen Wei, Konstantinos Psounis, and Kaipeng Zhang. TIR-Bench: A comprehensive benchmark for agentic thinking-with-images reasoning, 2025a. URL https://arxiv.org/abs/2511.01833.

Shilong Li, Xingyuan Bu, Wenjie Wang, Jiaheng Liu, Jun Dong, Haoyang He, Hao Lu, Haozhe Zhang, Chenchen Jing, Zhen Li, Chuanhao Li, Jiayi Tian, Chenchen Zhang, Tianhao Peng, Yancheng He, Jihao Gu, Hui Huang, Donghao Zhou, Yuanxing Zhang, Jian Yang, Ge Zhang, Wenhao Huang, Zhaoxiang Zhang, Qiangpeng Yang, and Shilei Wen. MM-BrowseComp: A comprehensive benchmark for multimodal browsing agents, 2025b. URL https://arxiv.org/abs/2508.13186.

Changzhi Liu, Yilun Liu, Sikuan Yan, Volker Tresp, and Yunpu Ma. Mendel gödel machine: Recursive self-improving coding agents via comparative evolution, 2026. URL https://arxiv.org/abs/2608.07645.

Lance Martin. Agent harness design: 3 patterns for harnessing Claude's intelligence. Anthropic, April 2026. URL https://claude.com/blog/harnessing-claudes-intelligence.

Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Tiancheng Han, Botian Shi, Wenhai Wang, Junjun He, Kaipeng Zhang, Ping Luo, Yu Qiao, Qiaosheng Zhang, and Wenqi Shao. MM-Eureka: Exploring the frontiers of multimodal reasoning with rule-based reinforcement learning, 2025. URL https://arxiv.org/abs/2503.07365.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery, 2025. URL https://arxiv.org/abs/2506.13131.

OpenAI. GPT-4o system card, 2024a. URL https://arxiv.org/abs/2410.21276.

OpenAI. OpenAI o1 system card, 2024b. URL https://arxiv.org/abs/2412.16720.

OpenAI. Introducing GPT-5 for developers, 2025. URL https://openai.com/index/introducing-gpt-5-for-developers/.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Zhaochen Su, Peng Xia, Hangyu Guo, Zhenhua Liu, Yan Ma, Xiaoye Qu, Jiaqi Liu, Yanshu Li, Kaide Zeng, Zhengyuan Yang, Linjie Li, Yu Cheng, Heng Ji, Junxian He, and Yi R. Fung. Thinking with images for multimodal reasoning: Foundations, methods, and future frontiers, 2025. URL https://arxiv.org/abs/2506.23918.

Zhaochen Su, Jincheng Gao, Hangyu Guo, Zhenhua Liu, Lueyang Zhang, Xinyu Geng, Shijue Huang, Peng Xia, Guanyu Jiang, Cheng Wang, Yue Zhang, Yi R. Fung, and Junxian He. AgentVista: Evaluating multimodal agents in ultra-challenging realistic visual scenarios, 2026. URL https://arxiv.org/abs/2602.23166.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7080–7106, Rabat, Morocco, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.eacl-long.333. URL https://aclanthology.org/2026.eacl-long.333/.

Xiangru Tang, Tianrui Qin, Tianhao Peng, Ziyang Zhou, Daniel Shao, Tingting Du, Xinming Wei, Peng Xia, Fang Wu, He Zhu, Ge Zhang, Jiaheng Liu, Xingyao Wang, Sirui Hong, Chenglin Wu, Hao Cheng, Chi Wang, and Wangchunshu Zhou. Agent KB: Leveraging cross-domain experience for agentic problem solving, 2025. URL https://arxiv.org/abs/2507.06229.

Xijia Tao, Yihua Teng, Xinxing Su, Xinyu Fu, Jihao Wu, Chaofan Tao, Ziru Liu, Haoli Bai, Rui Liu, and Lingpeng Kong. MMSearch-Plus: Benchmarking provenance-aware search for multimodal browsing agents. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2508.21475.

Muxin Tian, Zhe Wang, Blair Yang, Zhenwei Tang, Kunlun Zhu, Honghua Dong, Hanchen Li, Xinni Xie, Guangjing Wang, and Jiaxuan You. SWE-Bench Mobile: Can large language model agents develop industry-level mobile applications?, 2026. URL https://arxiv.org/abs/2602.09540.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-Agents enhances large language model capabilities. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5434be94e82c54327bb9dcaf7fca52b6-Paper-Conference.pdf.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911. PMLR, 2025b. URL https://proceedings.mlr.press/v267/wang25bx.html.

Feng Xiong, Zengbin Wang, Yong Wang, Xuecai Hu, Jinghan He, Liang Lin, Yuan Liu, and Xiangxiang Chu. Ace-Skill: Bootstrapping multimodal agents with prioritized and clustered evolution, 2026. URL https://arxiv.org/abs/2605.08887.

Huanjin Yao, Jiaxing Huang, Wenhao Wu, Jingyi Zhang, Yibo Wang, Shunyu Liu, Yingjie Wang, Yuxin Song, Haocheng Feng, Li Shen, and Dacheng Tao. Mulberry: Empowering MLLM with o1-like reasoning and reflection via collective monte carlo tree search, 2024. URL https://arxiv.org/abs/2412.18319.

Huanjin Yao, Qixiang Yin, Min Yang, Ziwang Zhao, Yibo Wang, Haotian Luo, Jingyi Zhang, and Jiaxing Huang. MM-DeepResearch: A simple and effective multimodal agentic search baseline, 2026. URL https://arxiv.org/abs/2603.01050.

Haoran Ye, Xuning He, Vincent Arak, Haonan Dong, and Guojie Song. Meta context engineering via agentic skill evolution, 2026. URL https://arxiv.org/abs/2601.21557.

Licheng Yu, Patrick Poirson, Shan Yang, Alexander C. Berg, and Tamara L. Berg. Modeling context in referring expressions. In *European Conference on Computer Vision*, 2016. URL https://arxiv.org/abs/1608.00272.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative AI by backpropagating language model feedback. *Nature*, 639:609–616, 2025. doi: 10.1038/s41586-025-08661-4. URL https://www.nature.com/articles/s41586-025-08661-4.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-Harness: Harnesses that improve themselves, 2026. URL https://arxiv.org/abs/2606.09498.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin gödel machine: Open-ended evolution of self-improving agents, 2025a. URL https://arxiv.org/abs/2505.22954.

Jingyi Zhang, Jiaxing Huang, Huanjin Yao, Shunyu Liu, Xikun Zhang, Shijian Lu, and Dacheng Tao. R1-VL: Learning to reason with multimodal large language models via step-wise group relative policy optimization, 2025b. URL https://arxiv.org/abs/2503.12937.

Ziwei Zheng, Michael Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and Xing Yu. DeepEyes: Incentivizing thinking with images via reinforcement learning, 2025. URL https://arxiv.org/abs/2505.14362.

A HARNESS AND BENCHMARK DETAILS

A.1 TASK-AGENT HARNESS AND TOOLS

Our initial task-agent's harness follows XSkill (Jiang et al., 2026), exposing tools including Web Search, Image Search, Visit, and Code. The available subset depends on the benchmark as shown in Tab. 4 summarizes the tools in the initial harness, detailing their functionalities as well as their input and output interfaces.

Tool	Function	Inputs
Web Search	Retrieves ranked webpages with titles, URLs, and snippets.	`query`: search text; `max_results`: result limit (schema default: 10).
Image Search	Finds images from a text query or performs reverse image retrieval.	`search_type`: `text` or `reverse`; `query`: text query; `image_url`: image reference; `max_results`: result limit (schema default: 10).
Visit	Reads a retrieved webpage and extracts information relevant to a specified goal.	`url`: webpage address; `goal`: information to extract.
Code	Runs Python for image manipulation, numerical computation, and analysis in a persistent kernel.	`code`: Python source.and functions remain available across calls within a rollout.

Table 4: Initial task-agent tool interfaces. Schema defaults may be overridden by runtime configuration.

Optimization-agent harness. The optimization agent interacts with the harness workspace through Gemini CLI (Google, 2025) and uses file-operation and command-execution tools to inspect and modify the current harness. Specifically, it can read and edit files, locate relevant code, and execute commands to evaluate different candidate modifications. At each evolution round, the optimization agent analyzes the execution trajectories and feedback produced by the task agent, identifies potential issues in the current harness, and accordingly updates the task-agent harness. In MemCoHE, we additionally maintain a cross-round meta-memory, allowing the optimization agent to reuse experience accumulated throughout the evolution process. In this way, the task-agent tools are primarily responsible for solving individual tasks, whereas the optimization agent's development tools are used to analyze, modify, and iteratively improve the task-agent harnesses, with the two serving distinct but complementary roles in the overall evolution process.

Image Operation. The task agent can directly access the input image, where the original image is referenced by `original_image`, while intermediate images generated during tool execution are referenced using identifiers such as `tool_image_1`. The Python environment provides commonly used libraries including PIL, NumPy, OpenCV, Matplotlib, SciPy, scikit-learn, Pandas, and SymPy for image processing, numerical computation, and result analysis. Generated images can either be returned directly through `plt.show()` or saved for use in subsequent steps. Since the execution environment preserves intermediate states, the agent can carry out multi-step operations such as cropping, segmentation, and measurement without repeatedly reloading or passing the images between steps.

A.2 DETAILS OF BENCHMARKS

We follow the same training/test split as XSkill (Jiang et al., 2026), using random seed 42 for data partitioning. The training tasks are used for harness evolution, while the test tasks are used to evaluate the final selected harness. Tab. 5 summarizes the number of tasks used in our experiments and the tools available for each benchmark. Note that the reported task counts correspond to the

evaluation split used in our experiments and do not necessarily reflect the full size of the latest version of each public benchmark.

Dataset	Train	Test	Code	Web Search	Img Search	Visit
Visual Tool-Based Reasoning						
VisualToolBench	100	214	✓	✓	–	✓
TIR-Bench	100	200	✓	–	–	–
Multimodal Search and Browsing						
MMSearch-Plus	100	211	✓	✓	✓	✓
MMBrowseComp	–	130	✓	✓	✓	✓
Comprehensive Real-World Tasks						
AgentVista	100	109	✓	✓	✓	✓

Table 5: Benchmark splits and tool availability. Train/Test denote task counts. MMBrowseComp is used only for transfer evaluation.

VisualToolBench. *VisualToolBench* (Guo et al., 2025) evaluates whether an agent can solve visual reasoning tasks by manipulating images and using tools. In many cases, the answer cannot be obtained directly from the original image, so the agent needs to crop, transform, or enhance the image before extracting the relevant evidence. Following XSkill, we use the single-turn tasks from the `hybrid_tool_reasoning` subset, with 100 tasks for harness evolution and 214 tasks for testing. Here, "single-turn" only means that the user provides one query; solving the task may still require multiple tool calls.

TIR-Bench. *TIR-Bench* (Li et al., 2025a) evaluates visual reasoning with image-processing tools. The full benchmark contains 13 task categories. Following XSkill, we keep the five categories that are compatible with our tool set: `refcoco`, `maze`, `instrument`, `ocr`, and `contrast`. They cover referring-expression understanding, spatial reasoning, object recognition, text reading, and image comparison. This gives 430 samples in total, of which 100 are used for harness evolution and 200 for testing.

MMSearch-Plus. *MMSearch-Plus* (Tao et al., 2026) evaluates whether an agent can use visual clues from the input image to search for information beyond the image itself. Solving these tasks often requires several rounds of image and text search, as well as checking which retrieved results are actually relevant, since similar images and noisy search results can be misleading. The benchmark contains 311 tasks. We use 100 for harness evolution and the remaining 211 for testing.

MMBrowseComp. *MMBrowseComp* (Li et al., 2025b) evaluates multimodal web browsing. The agent needs to understand the visual input, search and browse external webpages, and combine evidence from different sources to answer the question. Following XSkill, we use 130 tasks as the test set and do not perform any additional harness evolution on this benchmark. Instead, we directly apply the harness evolved on *MMSearch-Plus* to *MMBrowseComp*, which allows us to evaluate how well the evolved harness transfers to an unseen benchmark.

AgentVista. *AgentVista* (Su et al., 2026) evaluates agents on more complex tasks that require the combined use of visual understanding, search, web browsing, and code execution. These tasks often involve fine-grained visual evidence and long sequences of tool interactions before the final answer can be reached. Following XSkill, we use 209 tasks in total, with 100 for harness evolution and 109 for testing.

A.3 Initial Task-Agent System Prompts

Detailed prompt design for Co-Harness.

CoT

```
INSTRUCTIONS:
1.  Analyze: Carefully observe the image and the user's question.
2.  Think: Explain your step-by-step reasoning process.
3.  Answer: Once you are confident in your findings, you MUST provide the
   final answer inside <answer> ...(your final answer)... </answer> tags!
    E.g., <answer>The answer is 10.</answer> / <answer> A </answer> / ...
```

Task Agent

```
You are a visual reasoning agent. Your goal is to answer questions about
   images.

# AVAILABLE TOOLS:

You have access to the following tools:
1. **web_search**: Search the web for information, facts, or current events
2. **image_search**: Search for related images using text query or reverse
   image search
3. **visit**: Visit a webpage and extract its main content
4. **code_interpreter**: Execute Python code for image processing,
   analysis, and calculations

# INSTRUCTIONS:

1. Analyze: Carefully observe the image and the user's question.
2. Think: Explain your step-by-step reasoning process.
3. Use Tools: Call appropriate tool to gather information and help answer
   the question.
4. Iterate as needed: Continue reasoning and using tools in next turns
   until you are confident in your findings.
5. Answer: Once confident, provide the final answer inside
   <answer>...</answer> tags!

# IMPORTANT:

- Always explain your detailed reasoning process before using any tool.
- You can ONLY call one tool at one turn! Do not call multiple tools in
   one turn!
- You MUST provide your final answer using complete <answer>...</answer>
   tags!
```

Router Agent

```
You are the routing component of a cooperative multimodal agent system.
Classify the task by the operations and capabilities it requires, then
   select
exactly one of the available harnesses. You have no tools and must not
   solve the
task. Every listed harness is a valid route. Return only JSON in this form:
{"harness":"harness_id","reason":"brief capability match"}

Prefer the harness whose description best matches the task. Never invent
   an ID.
```

Setting	AWM	DC	Agent-KB
Retrieved items (k)	1	1	3
Memory representation	Workflow	Cheatsheet	Dual guidance
Memory length limit	200 words	200 words	Not specified
Retrieval	Semantic	Semantic	Lexical/semantic (0.5/0.5)
Online memory revision	No	Yes	No
Knowledge construction	LLM induction	Synthesis	Query reasoning
Success-only filtering	Yes	No	Yes

Table 6: Baseline reference configurations.

B Baseline and Evaluation Details

B.1 Baseline Details

Reasoning with CoT and Tools. The CoT baseline answers using the model's reasoning without external tool calls. The Tools baseline uses the initial task-agent harness and the benchmark's permitted tools, without harness evolution. Their system prompt templates are given in Appendix A.3.

Agent Workflow Memory (AWM). AWM (Wang et al., 2025b) turns prior task executions into reusable workflows. Retrieved workflows supply procedural guidance for subsequent tasks, allowing the agent to reuse established action sequences.

Dynamic Cheatsheet (DC). DC (Suzgun et al., 2026) maintains a compact memory of useful strategies, code fragments, and lessons from previous executions. The memory is revised as new tasks are processed, allowing knowledge to accumulate without updating model parameters.

Agent-KB. Agent-KB (Tang et al., 2025) organizes experience for reuse across tasks and agent systems. Retrieval supplies planning guidance before execution and diagnostic guidance when difficulties arise.

XSkill. XSkill (Jiang et al., 2026) maintains complementary stores of task-level skills and action-level experience. It summarizes execution trajectories, compares outcomes, and consolidates reusable guidance. At inference time, task decomposition supports retrieval, followed by adaptation of the retrieved knowledge to the current visual context. The task agent executes this guidance through its tool-enabled harness.

Meta-Harness. Meta-Harness (Lee et al., 2026) searches over harness code using an optimization agent. The proposer can inspect earlier candidates, evaluation scores, and execution traces stored in the workspace. In our comparison, it evolves a single shared harness, providing a harness-level baseline against which to evaluate the mixture of expert harnesses and collective learning in Co-Harness.

B.2 Judging and Metrics

Final answers are scored by GPT-4o (OpenAI, 2024a) using the corresponding benchmark's judging criteria and reference answer. Each rollout receives a binary result: 1 for a correct answer and 0 otherwise. We report Average@4, the mean success rate over four independent rollouts per task, and Pass@4, the fraction of tasks with at least one successful rollout. The same evaluation protocol is applied to all compared methods. Test-set results are not used to select the evolved harness.

C Co-Harness Configuration

C.1 Evolution and Inference Settings

Tab. 7 summarizes the settings used in the main experiments. During evolution, each candidate harness is evaluated once per training task. We retain the candidate with the best training-set performance across eight evolution rounds and use it for test-time execution. Model parameters remain fixed throughout.

Setting	**Value**
Temperature	1.0
Top-p	1.0
Maximum interaction turns	20
Maximum tokens per turn	8,192
Maximum input images	100
Evolution rounds	8
Initial domains	3
Initial experts per domain	2
Rollouts per training task per candidate	1
Independent rollouts per test task	1
Harness selection	Best training-set performance

Table 7: Co-Harness evolution and inference settings.