VISEVOLVER: SELF-EVOLVING MULTIMODAL AGENTS BY ROUTING EXPERIENCE INTO MEMORY AND PARAMETERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal agents inspect images, invoke tools, and reason over the observations they acquire, yet the experience produced by these interactions rarely outlives the task that produced it. External Memory makes an experience immediately reusable while leaving the policy unchanged, and parameter updates make behavior permanent while discarding the context that justified it. Treating the two as alternatives skips the prior question of which experience deserves which form of persistence. We present **VisEvolver**, a hybrid Parameter-and-Memory self-evolving framework that answers this question inside one experience pipeline. Causal Experience Attribution audits every candidate through its task outcome, its executed actions, and a set of paired visual interventions, so that only behavior whose benefit traces back to identifiable image content is retained. An Experience Router then reads cross-task support, mean utility, utility stability, instance specificity, and conflict, and sends each audited experience to editable process Memory, to the model parameters, or to both. Experience admitted for internalization is learned through Student-Reachable Hindsight Supervision, which supervises validated actions only at states the deployed policy actually visits and protects already-correct behavior with a preservation term. Across chart reasoning, infographic understanding, visual tool use, and embodied interaction on two backbones, VisEvolver improves over the base agent in every setting and attains the strongest score in seven of eight. Neither branch alone reproduces the complete system, which shows that explicit experience reuse and selective parameter learning carry different and additive information[1].

1 INTRODUCTION

Multimodal large language models have moved from describing images to acting on the world, choosing which tool to invoke, which region to inspect, and when the evidence in hand is sufficient to answer (Liu et al., 2023; Li et al., 2023; Wang et al., 2024). Visual programming systems compose perception and computation into executable steps (Gupta & Kembhavi, 2023; Surís et al., 2023), subsequent work turns revisited image regions into part of the reasoning trace (Hu et al., 2024; Wang et al., 2026), and recent agents learn when a visual operation is worth performing at all (Zheng et al., 2026; Hong et al., 2026). The same interaction pattern now reaches browsers, operating systems, and embodied environments (He et al., 2024; Xie et al., 2024; Xu et al., 2025), where every episode leaves behind observations, executed actions, verifier feedback, and a record of which decisions worked. However, almost all of this experience is discarded the moment a task ends, and the agent meets the next task with exactly the capability it already had.

An agent that keeps this experience must first decide what a successful episode actually taught it. One trajectory interleaves behavior that transfers, such as enlarging a dense axis label before reading it, with content that never will, such as the value printed on one particular chart. Appearing inside a correct trajectory is weak evidence of contribution, and our paired interventions show that a randomly chosen region of the same image recovers about half of the measured benefit on chart reasoning, so a pipeline that retains whatever accompanied a correct answer would fill most of its

[1]Code are available in supplementary materials.

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.

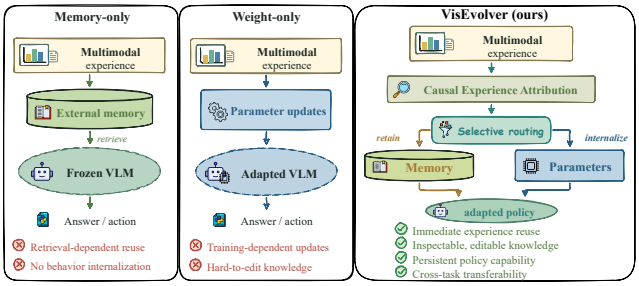

Figure 1: Memory-only self-evolution keeps experience beside an unchanged policy, and Weight-only self-evolution writes experience into the policy and loses its context. VisEvolver audits experience for causal visual support and routes it to editable process Memory, to the parameters, or to both, according to how far its benefit is shown to travel.

repository with content that was merely present. Useful experience also differs in how far it travels. A page state or a crop coordinate serves the current task alone, whereas checking conflicting visual evidence before answering applies to tasks the agent has not yet seen. What an episode taught and how far that lesson reaches are therefore two separate questions, and both must be answered before any experience is stored.

Existing work preserves multimodal experience along two routes. *External-memory methods* keep information beside the policy, retaining events, tips, and structured records for later tasks (Fan et al., 2024; Wang et al., 2025a; Liu et al., 2026; Zhang et al., 2026d). These records stay inspectable and editable, although the agent must re-read them on every task that needs them, and a retrieved record never reveals which of its observations made it useful. *Parameter-learning methods* instead write experience into the policy, training on autonomous interaction and distilling privileged perception into a full-image student (Bai et al., 2024; Sun et al., 2026; Qin et al., 2025; Yuan et al., 2026; Aniri et al., 2026; Zhao et al., 2026). These updates need no retrieval, although they absorb the behavior together with the context that justified it and cannot be revised afterwards. In essence, both lines answer how to persist an experience and leave unanswered which form of persistence a given experience has earned.

To this end, we propose **VisEvolver**, a hybrid Parameter-and-Memory self-evolving framework that makes this decision an explicit and evidence-driven step of the pipeline. As illustrated in Figure 1, VisEvolver derives both an editable process Memory and an adapted policy from the same audited experience, so the two forms of persistence compete for each candidate instead of being chosen once for the whole system. Specifically, to mitigate the weak evidence carried by co-occurrence, Memory-Grounded Experience Acquisition admits a candidate only when its task outcome, its executed actions, and a set of paired visual interventions all support it, where the interventions compare the candidate region with a text-only input, with same-image distractors, and with the region removed. To mitigate the confusion between a lesson and its instance, an Experience Router scores each audited record along cross-task support, mean utility, utility stability, instance specificity, and conflict, and directs it to Memory, to the parameters, or to both, so that narrowly supported records stay editable while repeatedly confirmed ones enter Selective Experience Internalization. To mitigate the mismatch between a teacher trace and a student policy, internalization uses Student-Reachable Hindsight Supervision, which supervises a validated action only at a state the deployed policy actually visits and never copies an answer that depends on an observation the policy has not yet acquired. A Base Preservation term holds already-correct behavior in place, and at inference one adapted policy reads the frozen Memory inside a single action-observation loop.

Across chart reasoning, infographic understanding, visual tool use, and embodied interaction with Qwen3.5-9B and Gemma-3-12B, VisEvolver improves over the base agent in all eight backbone-task settings and exceeds both single-branch variants throughout, while the two branches trade places between tasks in a way that tracks whether a task rewards an explicit procedure or a changed perceptual habit. Our contributions are threefold.

- **Hybrid Parameter-and-Memory Self-Evolution.** To the best of our knowledge, VisEvolver is the first self-evolving multimodal agent that decides the destination of an experience from evidence about its own benefit, producing an editable process Memory and an adapted policy from one audited experience stream.

- **Causally Attributed and Selectively Internalized Experience.** Causal Experience Attribution establishes visual support through paired interventions, the Experience Router turns utility evidence into a routing decision, and Student-Reachable Hindsight Supervision converts admitted experience into actions the deployed policy can execute.
- **Consistent Gains across Tasks and Backbones.** VisEvolver attains the strongest score in seven of eight backbone-task settings, and its component study shows that visual attribution, explicit procedures, and internalized behavior each repair a distinct failure mode.

2 Related Work

Multimodal Agents that Act on Visual Evidence. Multimodal agents extend visual understanding from passive interpretation to active evidence acquisition, on top of the image-language interface established by visual instruction tuning (Liu et al., 2023; Li et al., 2023; Wang et al., 2024). VISPROG (Gupta & Kembhavi, 2023) and ViperGPT (Surís et al., 2023) compose visual modules into executable programs, Visual Sketchpad (Hu et al., 2024) and VGR (Wang et al., 2026) promote intermediate visual artifacts into the reasoning trace, DeepEyes (Zheng et al., 2026; Hong et al., 2026) learns when inspecting an image is worthwhile, and the same pattern reaches browser and computer environments (He et al., 2024; Xie et al., 2024; Xu et al., 2025). This line establishes how an agent gathers evidence within a single task and leaves open which part of the resulting episode should change its behavior on the next one.

Experience Stored outside the Policy. One family keeps experience beside the model, organizing temporal events for long-video understanding (Fan et al., 2024), retaining tips and shortcuts for later mobile tasks (Wang et al., 2025a), carrying guidance across browsing sessions (Liu et al., 2026), and recombining structured records associatively (Zhang et al., 2026d). Explicit storage keeps knowledge auditable and revisable, although retrieving a relevant record does not reveal which observation made it useful, and every benefit is paid for again through retrieval on each task.

Experience Written into the Policy. A second family converts interaction into parameter updates, learning from autonomous device interaction (Bai et al., 2024; Sun et al., 2026) and iterating on collected interface traces (Qin et al., 2025). Recent multimodal distillation sharpens the supervision behind such updates, transferring crop-conditioned perception to a full-image student (Yuan et al., 2026), contrasting zoomed and masked teachers to select visually supported tokens (Aniri et al., 2026), and combining modality-specific teachers on student-generated responses (Zhao et al., 2026). These methods align parameter learning with student behavior, although they commit every admitted experience to the same destination and give up the ability to revise it afterwards.

Positioning. In this work, we target the decision that precedes both families, namely which experience should stay explicit and which should become part of the policy. Unlike previous approaches that fix one form of persistence for the whole system and then optimize inside it, VisEvolver first establishes the causal visual support of a candidate experience and then routes it according to the range of tasks over which its benefit has been confirmed, so that narrowly supported records remain editable in process Memory while records with repeated cross-task support additionally enter parameter learning under Student-Reachable Hindsight Supervision and Base Preservation.

3 VisEvolver

VisEvolver turns a stream of solved and failed multimodal tasks into two forms of lasting capability. The difficulty is that a trajectory records everything the agent saw and did, while only part of it caused the outcome and only part of that generalizes beyond the instance. VisEvolver therefore separates acquisition from consolidation. Memory-Grounded Experience Acquisition in Section 3.2 collects grounded interactions and establishes which observations and actions are causally supported, and Hybrid Parameter-and-Memory Self-Evolution in Section 3.3 routes each audited record to process Memory, to the model parameters, or to both.

3.1 Self-Evolution over a Stream of Multimodal Tasks

We consider a stream of multimodal tasks instead of isolated image-question pairs, because the value of an experience is only visible once later tasks can use it. At round t, a task $x_t = (I_t, q_t)$ contains

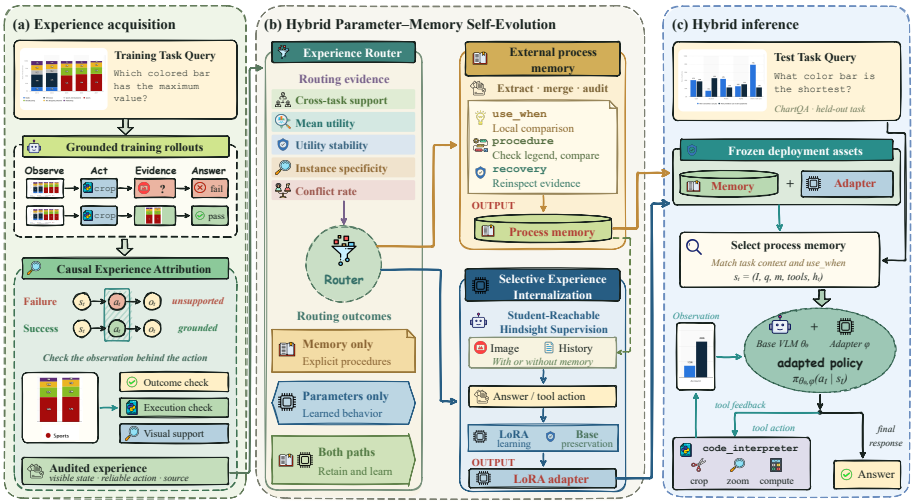

Figure 2: VisEvolver derives two complementary forms of persistence from one audited experience stream. (a) Grounded rollouts pass an outcome check, an execution check, and a visual-support check, so only experience with traceable causal support becomes admissible. (b) The Experience Router sends each admitted record to editable process Memory, to Selective Experience Internalization, or to both, and hindsight supervision reaches only states the Student itself visits. (c) One adapted policy then reads the frozen Memory inside a single action-observation loop.

one or more images I_t and a textual instruction q_t. The policy π_θ acts through a fixed tool set $\mathcal{T}$ and receives a textual or visual observation after each action, which yields a trajectory and a task-level outcome,

$$\tau_t = \{(s_{t,i}, a_{t,i}, o_{t,i+1})\}_{i=1}^{H_t}, \qquad R_t = f_{\text{score}}(\tau_t), \tag{1}$$

where $s_{t,i}$ denotes the state before action $a_{t,i}$ and holds the system instruction, the original images, the question, the retrieved Memory, the tool interfaces, and the interaction history, and $f_{\text{score}}(\cdot)$ denotes an answer checker or an environment verifier. Unlike standard evaluation, VisEvolver does not discard τ_t. It maintains an evolving repository $\mathcal{M}_t$ and treats every interaction as a source of both task behavior and a candidate repository update,

$$c_t = f_{\text{sel}}\big(x_t, f_{\text{ret}}(x_t, \mathcal{M}_t)\big), \qquad \tau_t \sim \pi_\theta(\cdot \mid x_t, c_t, \mathcal{T}), \qquad \mathcal{M}_{t+1} = \mathcal{M}_t \oplus \Delta_t, \tag{2}$$

where $f_{\text{ret}}(\cdot)$ denotes non-parametric retrieval, $f_{\text{sel}}(\cdot)$ denotes the policy's own selection of a compact context c_t from the retrieved candidates, Δ_t denotes the validated updates of this round, and $\oplus$ denotes insertion, merging, or revision. Each round should therefore solve its own task and should leave $\mathcal{M}_{t+1}$ and θ more capable than it found them, without overwriting behavior that already worked.

3.2 Memory-Grounded Experience Acquisition

The goal of this stage is to obtain experience whose usefulness can be defended, which requires separating observations that changed an outcome from observations that merely accompanied it. Since a plausible explanation written after the fact cannot make this separation, VisEvolver collects interactions under deployment conditions and audits them against the actual observations they produced, as shown in Figure 2(a). Policy parameters stay fixed throughout.

Three-Level Grounded Experience Repository The repository retains experience at the level of detail that each use requires, so that auditing, evidence reuse, and guidance need not compete for one representation,

$$\mathcal{M}_t = \mathcal{M}_t^{\text{epi}} \cup \mathcal{M}_t^{\text{evi}} \cup \mathcal{M}_t^{\text{proc}}, \qquad \mathcal{C}_t = \bigcup_{\ell} \text{TopK}_{m \in \mathcal{M}_t^{\ell}} \text{sim}\left(f_{\text{emb}}(x_t), f_{\text{emb}}(m)\right), \tag{3}$$

where $\mathcal{M}^{\text{epi}}$ denotes source episodes with their original context, $\mathcal{M}^{\text{evi}}$ denotes verified Visual Evidence, $\mathcal{M}^{\text{proc}}$ denotes reusable process Memory, and $f_{\text{emb}}(\cdot)$ denotes the shared encoder behind

retrieval. The policy selects c_t from the candidate set $\mathcal{C}_t$ by matching each procedure against the task and its applicability conditions, and then acts on the original input and its own observed history,

$$a_{t,i} \sim \pi_\theta\big(\cdot \mid x_t, c_t, o_{t,<i}, a_{t,<i}, \mathcal{T}\big). \tag{4}$$

Every tool call returns a real image or text observation that enters the next state, and retaining these observations is what later allows an audit to tell a supported decision from a convincing narrative.

Causal Experience Attribution through Paired Visual Interventions A correct answer certifies the trajectory and not the procedure inside it, so each completed episode passes three audits. An outcome check confirms task success, an execution check confirms that the recorded action ran as written, and a visual-support check asks whether the observation behind a decision carried the information the decision used. For a candidate region r, we compare the candidate crop with a text-only input, with same-image distractor crops, and with the candidate replaced by a blank crop, which yields three attribution signals,

$$G_{\mathrm{roi}}(r) = S_{\mathrm{cand}} - S_{\mathrm{text}}, \qquad G_{\mathrm{spec}}(r) = S_{\mathrm{cand}} - \max_d S^{(d)}_{\mathrm{dist}}, \qquad D_{\mathrm{mask}}(r) = S_{\mathrm{cand}} - S_{\mathrm{blank}}, \tag{5}$$

where S_{cand}, S_{text}, $S^{(d)}_{\mathrm{dist}}$, and S_{blank} denote the task scores obtained with the candidate crop, with text alone, with the d-th distractor crop, and with the blanked crop. The three signals separate the value of seeing an image at all, the value of seeing this region instead of another, and the value of the content the region contains. A region becomes Visual Evidence only when all three agree, which rejects both the region that any crop could replace and the region whose content the model never read.

The audits also compare a successful trajectory against a failed one from the same task family, which localizes the decision worth reviewing at the first divergence that separates their outcomes,

$$j^* = \min\big\{\, j \;\big|\; a^+_j \neq a^-_j \;\wedge\; R(\tau^+) > R(\tau^-) \,\big\}. \tag{6}$$

The divergence at j^* nominates a candidate and never admits it. The action survives only when the execution audit and the visual signals of Eq. 5 agree, which keeps a transferable habit such as enlarging small text before reading it while discarding the literal text that one image happened to contain.

Confidence-Adjusted Experience Utility A single supporting task cannot distinguish a reusable lesson from a coincidence, so the audited evidence is aggregated into a utility that grows with independent corroboration,

$$V(e) = \gamma(e)\big[\alpha G_{\mathrm{visual}}(e) + \beta G_{\mathrm{behavior}}(e)\big], \qquad \gamma(e) = 1 - \frac{1}{\sqrt{1+n(e)}}, \tag{7}$$

where $n(e)$ denotes the number of independent tasks supporting the experience e, $G_{\mathrm{visual}}(e)$ summarizes the intervention signals, $G_{\mathrm{behavior}}(e)$ summarizes verified behavioral gains, and α and β balance the two sources. The factor $\gamma(e)$ keeps a strongly-rewarded single observation below a modest benefit confirmed many times. Conflicting or weakly-supported records stay available for diagnosis and enter neither process Memory nor parameter supervision.

3.3 Hybrid Parameter-and-Memory Self-Evolution

Audited experience can improve future behavior in two ways, and the goal of this stage is to give each record the form that fits the evidence behind it. An explicit procedure remains readable and revisable while costing retrieval on every use, and a parameter update needs no retrieval while becoming difficult to undo. VisEvolver therefore separates the routing decision from the two update paths, as shown in Figure 2(b).

Experience Router Deciding where an experience belongs requires evidence about how far its benefit travels, which no single utility score provides. The Router acts on each audited candidate, namely a procedure abstracted from one or more episodes together with the Visual Evidence that supports it, and reads five signals that each rule out a different way an experience can fail to transfer, namely cross-task support $n(e)$, mean utility $\bar{V}(e)$, utility variance $\mathrm{Var}(V(e))$, instance specificity

$s(e)$, and conflict rate $c(e)$. An experience is admitted for internalization only when all five clear their thresholds,

$$f_{\text{route}}(e) = \mathbb{I}\big[\, n(e) \geq \tau_n,\ \ \bar{V}(e) \geq \tau_V,\ \ s(e) \leq \tau_s,\ \ \text{Var}(V(e)) \leq \tau_v,\ \ c(e) \leq \tau_c \,\big],$$
$$f_{\text{keep}}(e) = \mathbb{I}\big[\, s(e) > \tau_s \ \ \vee \ \ \texttt{tool_constraints}(e) \neq \varnothing \,\big], \tag{8}$$

where each threshold states the support required or the risk tolerated before an experience may change parameters, so that a benefit depending on unnamed conditions, a lesson fused with its instance, or a record contradicting one already admitted is all held back. The retention criterion f_{keep} is deliberately independent, since an experience too specific to internalize is exactly the one an explicit procedure should carry. The two criteria yield the three destinations of Figure 2(b), where Memory-only keeps an editable procedure, Parameters-only learns a behavior without adding a procedure to inference Memory, and Both retains and learns it. The source episode in $\mathcal{M}^{\text{epi}}$ stays traceable in every case, so a wrong admission remains reversible.

Instance-Free Process Memory A record that still contains its own answer teaches the next task nothing and can actively mislead it, so the writer strips instance-bound answers, URLs, coordinates, file paths, and incidental tool arguments, and keeps only what a later task can act on,

$$e = \big(\texttt{use_when},\ \texttt{procedure},\ \texttt{tool_constraints},\ \texttt{recovery},\ \texttt{provenance}\big), \tag{9}$$

where `use_when` states the task and evidence conditions under which the procedure applies, `recovery` states how to detect failure and recheck the evidence, and `provenance` links the compact record back to its source episodes. Two records merge only when both their conditions and their actions are compatible, which keeps a procedure actionable without promoting one answer into a general rule.

Selective Experience Internalization by Student-Reachable Hindsight Copying a teacher trace into the policy teaches actions the policy cannot reach, because hindsight lets a teacher rely on observations the student has not yet acquired. Internalization therefore runs in the student's own state distribution. The Student first executes the task with deployment inputs, tools, and the intended Memory setting, and the Teacher inspects the admitted experience and proposes a correction at a state the Student actually visited, whose target must be executable from the image, history, tools, and Memory available at that state. We retain a validated Teacher action when its continuation improves the verified outcome,

$$\mathcal{D}_{\text{hind}} = \big\{ (s_j^S, a_j^T, V(e_j)) \,\big|\, R(\tau^T) > R(\tau^S),\ \ V(e_j) > 0,\ \ f_{\text{route}}(e_j) = 1 \big\}, \tag{10}$$

where s_j^S denotes a Student-visited state and a_j^T denotes the first validated recovery action at that state, which may be an answer or a tool call. When the Student answers before inspecting a small region and the Teacher first crops that region, the target at the Student's state is the crop and not the value later read from it, so the supervision teaches what to do at a reachable decision point instead of reproducing a finished trace.

Base Preservation Learning a new behavior can silently cost an existing one, and an agent that acquires a tool habit while losing a correct direct answer has not improved. We therefore optimize the token-level likelihood of the retained actions together with a penalty on states the base policy already handles,

$$\mathcal{L} = \mathcal{L}_{\text{hind}} + \lambda_{\text{pres}}\, D_{\text{KL}}\big(p_{\theta_0}(\cdot \mid s) \,\|\, p_\theta(\cdot \mid s)\big), \tag{11}$$

where $\mathcal{L}_{\text{hind}}$ denotes the negative log-likelihood of the actions in $\mathcal{D}_{\text{hind}}$ weighted by their utility, θ_0 denotes the base policy, and λ_{pres} controls how strongly already-correct behavior is held in place. LoRA (Hu et al., 2022) updates the language-side policy while the vision encoder stays frozen, which confines internalization to how the agent uses perception instead of how it perceives. Both the resulting adapter and the reviewed process Memory are frozen before evaluation.

3.4 Inference with Frozen Memory and Internalized Behavior

At deployment the two forms of persistence must act through one policy, otherwise a gain could come from an ensemble instead of from evolution. For a new task the retriever forms candidates, the same policy selects a compact procedural context, and the same policy emits either a final answer or a tool action,

$$a_i \sim \pi_\theta\big(\cdot \mid I, q, f_{\text{ret}}(q, \mathcal{M}), \mathcal{T}, h_i\big), \tag{12}$$

where h_i denotes the interaction history and each tool call appends its real text or image observation to h_i before the policy continues. Memory is read-only throughout held-out evaluation, and no test trajectory writes Memory or updates parameters. To attribute any gain to the intended source, we evaluate three configurations that share components pairwise,

$$\pi_{\text{Memory}} = \pi_{\theta_0}(\cdot \mid x, \mathcal{M}, \mathcal{T}), \qquad \pi_{\text{Weight}} = \pi_{\theta}(\cdot \mid x, \varnothing, \mathcal{T}), \qquad \pi_{\text{Full}} = \pi_{\theta}(\cdot \mid x, \mathcal{M}, \mathcal{T}). \quad (13)$$

Memory-only and Full read the same repository, and Weight-only and Full run the same adapted policy, so a difference between Full and either variant isolates the contribution of the branch that variant removes.

4 Experiments

Our experiments examine whether VisEvolver improves multimodal task performance, whether causal visual attribution identifies useful visual evidence, and how the main components contribute to the complete system.

4.1 Experimental Setup

Benchmarks. We evaluate on four tasks that stress different uses of experience. ChartQA (Masry et al., 2022) and InfographicVQA (Mathew et al., 2022) require reading dense visual detail and reward a perceptual habit such as inspecting a region before answering. VisualToolBench (Guo et al., 2025) requires composing image operations and rewards an explicit procedure, and EB-ALFRED (Yang et al., 2025) requires multi-step embodied interaction under environment feedback.

Baselines. The Base Agent runs the same backbone with the same tool set and the same maximum number of steps while persisting nothing between tasks, and it is the reference against which every form of self-evolution is read. Memory-based methods include AWM (Wang et al., 2025b), Dynamic Cheatsheet (Suzgun et al., 2026), AgentKB (Tang et al., 2025), ReasoningBank (Ouyang et al., 2026), and MemEvolve (Zhang et al., 2026a), and parameter-based self-evolution is represented by OPD-Evolver (Zhang et al., 2026b). We further report supervised fine-tuning, GRPO (Shao et al., 2024), and MemRL (Zhang et al., 2026c) as direct parameter-training references, and include the two single-branch variants of Eq. 13.

Implementation Details. We instantiate VisEvolver on Qwen3.5-9B and Gemma-3-12B. Internalization updates the language-side policy with LoRA (Hu et al., 2022) and keeps the vision encoder frozen, so any change reflects how the agent uses perception. Experience is acquired only from the training split of each task, and every comparison is made on held-out tasks where Memory is read-only and no trajectory updates parameters. Appendix A lists the remaining configurations.

4.2 Main Results

As reported in Table 1, VisEvolver attains the strongest score in seven of eight backbone-task settings, while neither existing family of self-evolution is strong across the board. On chart and infographic reasoning, memory-based self-evolution falls behind the base agent on several settings even though it retains far more information than it started with. These tasks require inspecting the relevant image region, while another chart's layout can compete with the current image for context. Parameter-based self-evolution avoids this interference and improves clearly, which confirms that the useful lesson here is a perceptual habit and not a text snippet. VisEvolver keeps that habit and still benefits from procedural guidance, and the margin it adds over its stronger single branch is largest on Gemma-3-12B, whose base agent leaves more perceptual mistakes available to repair.

The picture reverses on visual tool use and embodied interaction, where the reusable content is an ordered procedure together with the conditions under which it applies. Memory-only is weakest on visual tool use, where inapplicable procedures can trigger unnecessary tools and internalization restores reliable composition. MemEvolve does attain the best embodied success rate on Qwen3.5-9B, which we attribute to the small set of household routines this environment repeats, and the same method is among the weakest on every task that requires reading an image. The branch that wins therefore tracks whether the task rewards an explicit procedure or a changed perceptual habit, so a single destination cannot be fixed once for a system that faces both kinds of task.

Table 1: VisEvolver attains the strongest score in seven of eight backbone-task settings, while single-destination self-evolution improves one capability and degrades another. Rows are grouped as memory-based methods, parameter-based methods, and the two single-branch variants of VisEvolver. The best result of each block is in **bold** and the second best is underlined.

Method	ChartQA Relaxed Acc.	InfographicVQA ANLS	VisualToolBench Score	EB-ALFRED Success Rate
		Qwen3.5-9B		
Base Agent	59.38	61.94	9.46	8.00
AWM	50.62	64.22	9.35	6.50
Dynamic Cheatsheet	52.62	62.32	9.23	9.42
AgentKB	48.62	62.13	9.00	8.17
ReasoningBank	49.00	61.66	9.00	8.72
MemEvolve	46.73	58.74	9.00	**11.37**
OPD-Evolver	67.00	68.38	9.23	8.33
Memory-only	68.38	68.76	6.31	9.58
Weight-only	68.38	65.30	9.51	9.75
VisEvolver	**68.88**	**69.00**	**9.70**	10.58
		Gemma-3-12B		
Base Agent	44.00	42.95	3.27	2.42
AWM	47.00	39.61	3.39	2.75
Dynamic Cheatsheet	42.12	42.98	3.39	2.08
AgentKB	42.75	41.12	2.22	4.50
ReasoningBank	41.00	25.81	4.23	2.04
MemEvolve	32.00	29.03	3.74	2.50
OPD-Evolver	40.50	35.12	3.74	5.00
Memory-only	47.50	40.39	3.39	12.00
Weight-only	47.25	46.01	4.21	13.33
VisEvolver	**48.25**	**54.52**	**6.07**	**15.33**

Since VisEvolver ends with an updated policy, a further question is whether the audited pipeline earns its place over training on those interactions directly. We train the same backbone with supervised fine-tuning, GRPO, and MemRL on VisEvolver's experience stream and evaluate both tasks with direct answer supervision. As shown in Table 2, each direct-training method advances at most one capability. GRPO reaches the strongest infographic score of the four while its chart reasoning stays close to the base agent, and MemRL falls below the base agent on chart reasoning, where rewarding trajectories that used Memory encourages retrieval when the image already contains the answer. The pattern follows from optimizing an outcome signal that cannot distinguish an answer reached by inspecting the right region from one reached by chance. Routing the same interactions through attribution and selective internalization advances both capabilities at once and leaves behind a repository open to inspection.

Table 2: Direct parameter training advances one capability at a time, while routed experience advances both together. Chart reasoning is scored by relaxed accuracy and infographic understanding by ANLS. The best score of each column is in **bold** and the second best is underlined.

Method	ChartQA	InfographicVQA
SFT	62.50	67.82
GRPO	63.00	**70.92**
MemRL	41.50	57.71
VisEvolver	**68.88**	69.00

4.3 What Makes a Visual Region Worth Remembering?

We test whether the visual-support check identifies useful content by comparing the attributed region with four controlled substitutes. The substitutes remove the image entirely, keep a randomly chosen region of the same image at the same size, insert a region from an unrelated image, and blank the attributed region while keeping its position and size. Figure 3 reports the outcome on chart and infographic reasoning.

A randomly chosen region of the same image recovers a substantial share of the benefit of the attributed region, which means that retaining whatever accompanied a correct answer would admit many regions whose content the agent never used. A region taken from an unrelated image recovers far less, so the benefit is not explained by the mere presence of an extra crop. Blanking the attributed region while preserving its geometry removes nearly all of the benefit, which places the causal content inside the region and not in its location. Requiring all three signals separates evidence from coincidence, and omitting this requirement produces weaker Memory on both tasks.

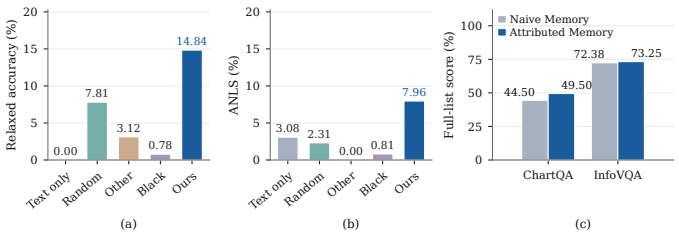

Figure 3: Paired interventions separate three sources of visual benefit. Bars report the task score obtained under each substitute, by relaxed accuracy on ChartQA and ANLS on InfographicVQA. A same-image random region recovers a substantial share of the gain, so co-occurrence with a correct answer is weak evidence of causal support, while blanking the attributed region removes almost all of it.

4.4 Which Design Choices Carry the Gain?

To verify that the routing decision itself carries information, we compare the complete system against a variant that keeps every other component and sends all admitted experience to both destinations, so that any difference isolates the value of deciding where an experience belongs. As shown in Table 3, this removal drops visual tool use to nearly the level of the variant that never internalizes anything. Indiscriminate internalization loses most of this gain by absorbing instance-specific tool sequences whose applicability conditions can no longer be inspected. The same removal costs far less on infographic reasoning, where most admitted records describe a perceptual habit that generalizes and the gate would have accepted them anyway.

Table 3: Removing the Router leaves tool composition near the level of no internalization at all, and no component can be recovered by the others. Columns report ANLS on InfographicVQA, the benchmark score on VisualToolBench, and success rate on EB-ALFRED with Qwen3.5-9B. The complete system is in **bold**.

Variant	Infographic	Tools	Embodied
VisEvolver	**69.00**	**9.70**	**10.58**
w/o Memory	65.30	9.51	9.75
w/o Internalization	68.76	6.31	9.58
w/o Attribution	66.87	8.88	9.21
w/o Router	67.40	6.54	10.00

We remove each remaining component to locate the losses it prevents. Removing Memory costs most on infographic reasoning, where an explicit procedure supplies the reading order that parameters alone do not fix, and removing internalization costs most on visual tool use, which matches the reading that reliable tool composition must become part of the policy. Removing visual attribution degrades every task at once, since the two branches then consume the same unfiltered stream. No component is therefore redundant with another.

5 Conclusion

In this work, we revisit what a multimodal agent should do with the experience it produces, and we argue that the question of where an experience should persist precedes the question of how to persist it. We present VisEvolver, a hybrid Parameter-and-Memory self-evolving framework that audits each candidate experience for causal visual support through paired interventions, routes it to editable process Memory, to the model parameters, or to both according to the range of tasks over which its benefit has been confirmed, and internalizes admitted experience only at states the deployed policy actually visits while holding already-correct behavior in place. Across four tasks and two backbones, VisEvolver improves over the base agent in every setting and exceeds both single-destination variants throughout, and the branch that dominates changes with whether a task rewards an explicit procedure or a changed perceptual habit. The component study further shows that a randomly chosen region of the same image recovers much of the benefit attributed to a retained region, which makes causal attribution a prerequisite for any form of experience retention. Treating the destination of an experience as a decision to be earned therefore offers a practical route toward agents that accumulate capability instead of records.

AI USE STATEMENT

In this work, we used generative AI tools only for final-stage language polishing and grammar refinement of the manuscript. We have not used generative AI tools for generating research ideas, designing methods, conducting experiments, implementing code, analyzing data, interpreting results, or producing scientific claims, and other generative AI-assisted tasks requiring disclosure are not applicable to this work. We have reviewed all AI-assisted revisions to ensure their correctness and consistency with the authors' intended content. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Aniri, Jinhe Bi, Peng Liao, Zengjie Jin, Volker Tresp, Fei Shen, Yunpu Ma, and Tat-Seng Chua. OPD-V: Visual On-Policy Self-Distillation with Modality Balance. arXiv preprint arXiv:2608.05131, 2026. URL https://arxiv.org/abs/2608.05131.

Hao Bai, Yifei Zhou, Mert Cemri, Jiayi Pan, Alane Suhr, Sergey Levine, and Aviral Kumar. DigiRL: Training In-The-Wild Device-Control Agents with Autonomous Reinforcement Learning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 12461–12495. Curran Associates, Inc., 2024. doi: 10.52202/079017-0397. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/1704ddd0bb89f159dfe609b32c889995-Paper-Conference.pdf.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient Finetuning of Quantized LLMs. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 10088–10115. Curran Associates, Inc., 2023. doi: 10.52202/075280-0441. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1feb87871436031bdc0f2beaa62a049b-Paper-Conference.pdf.

Yue Fan, Xiaojian Ma, Rujie Wu, Yuntao Du, Jiaqi Li, Zhi Gao, and Qing Li. VideoAgent: A Memory-Augmented Multimodal Agent for Video Understanding. In *Computer Vision – ECCV 2024*, pp. 75–92. Springer Nature Switzerland, September 2024. ISBN 9783031726705. doi: 10.1007/978-3-031-72670-5_5. URL https://doi.org/10.1007/978-3-031-72670-5_5.

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. arXiv preprint arXiv:2510.12712, 2025. URL https://arxiv.org/abs/2510.12712.

Tanmay Gupta and Aniruddha Kembhavi. Visual Programming: Compositional Visual Reasoning Without Training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14953–14962, June 2023.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. WebVoyager: Building an End-to-End Web Agent with Large Multimodal Models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6864–6890, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.371. URL https://aclanthology.org/2024.acl-long.371/.

Jack Hong, Chenxiao Zhao, Chenglin Zhu, Weiheng Lu, Guohai Xu, and Xing Yu. DeepEyesV2: Toward Agentic Multimodal Model. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 114851–114872, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/badbef7d421640fb038399e9d7271b1c-Paper-Conference.pdf.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah Smith, and Ranjay Krishna. Visual Sketchpad: Sketching as a Visual Chain of Thought for Multimodal Language Models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 139348–139379. Curran Associates, Inc., 2024. doi: 10.52202/079017-4423. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/fb82011040977c7712409fbdb5456647-Paper-Conference.pdf.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19730–19742. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/li23q.html.

Genglin Liu, Shijie Geng, Sha Li, Hejie Cui, Sarah Zhang, Xin Liu, and Tianyi Liu. WebCoach: Improve Browsing Agents with Memory-Aware Guidance for Better Web Navigation. In *ICLR 2026 Workshop on Memory for LLM-Based Agentic Systems*, 2026. URL https://arxiv.org/abs/2511.12997.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual Instruction Tuning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 34892–34916. Curran Associates, Inc., 2023. doi: 10.52202/075280-1516. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6dcf277ea32ce3288914faf369fe6de0-Paper-Conference.pdf.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A Benchmark for Question Answering about Charts with Visual and Logical Reasoning. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 2263–2279, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.177. URL https://aclanthology.org/2022.findings-acl.177/.

Minesh Mathew, Viraj Bagal, Rubèn Tito, Dimosthenis Karatzas, Ernest Valveny, and C.V. Jawahar. InfographicVQA. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 1697–1706, January 2022.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. ReasoningBank: Scaling Agent Self-Evolving with Reasoning Memory. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 94327–94354, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/980ea04d23d1f6908964eba2a74afe45-Paper-Conference.pdf.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, Xiaojun Xiao, Kai Cai, Chuang Li, Yaowei Zheng, Chaolin Jin, Chen Li, Xiao Zhou, Minchao Wang, Haoli Chen, Zhaojian Li, Haihua Yang, Haifeng Liu, Feng Lin, Tao Peng, Xin Liu, and Guang Shi. UI-TARS: Pioneering Automated GUI Interaction with Native Agents. arXiv preprint arXiv:2501.12326, 2025. URL https://arxiv.org/abs/2501.12326.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of

Mathematical Reasoning in Open Language Models. arXiv preprint arXiv:2402.03300, 2024. URL https://arxiv.org/abs/2402.03300.

Zeyi Sun, Ziyu Liu, Yuhang Zang, Yuhang Cao, Xiaoyi Dong, Tong Wu, Dahua Lin, and Jiaqi Wang. SEAgent: Self-Evolving Computer Use Agent with Autonomous Learning from Experience. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/65711.

Dídac Surís, Sachit Menon, and Carl Vondrick. ViperGPT: Visual Inference via Python Execution for Reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11888–11898, October 2023.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic Cheatsheet: Test-Time Learning with Adaptive Memory. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7080–7106, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-380-7. 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, He Zhu, Ge Zhang, Jiaheng Liu, Xingyao Wang, Sirui Hong, Chenglin Wu, and Wangchunshu Zhou. Agent KB: A Hierarchical Memory Framework for Cross-Domain Agentic Problem Solving. In *ICML 2025 Workshop on Collaborative and Federated Agentic Workflows*, 2025. URL https://openreview.net/forum?id=ohXoWHlrn8. Extended version available as arXiv:2507.06229.

Jiacong Wang, Zijian Kang, Haochen Wang, Xiao Liang, Ya Wang, Jiawen Li, Bohong Wu, Jiao Ran, Haiyong Jiang, Chao Feng, and Jun Xiao. VGR: Visual Grounded Reasoning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 89506–89528, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/90aeee6dfe75ab6b5d6958bce40d3e16-Paper-Conference.pdf.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution. arXiv preprint arXiv:2409.12191, 2024. URL https://arxiv.org/abs/2409.12191.

Zhenhailong Wang, Haiyang Xu, Junyang Wang, Xi Zhang, Ming Yan, Ji Zhang, Fei Huang, and Heng Ji. Mobile-Agent-E: Self-Evolving Mobile Assistant for Complex Tasks. In *NeurIPS 2025 Workshop on Scaling Environments for Agents*, 2025a. URL https://openreview.net/forum?id=GRmrGws6Lf.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent Workflow Memory. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911. PMLR, 13–19 Jul 2025b. URL https://proceedings.mlr.press/v267/wang25bx.html.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 52040–52094. Curran Associates, Inc., 2024. doi: 10.52202/079017-1650. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5d413e48f84dc61244b6be550f1cd8f5-Paper-Datasets_and_Benchmarks_Track.pdf.

Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. Aguvis: Unified Pure Vision Agents for Autonomous GUI Interaction. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 69772–69805. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/xu25ae.html.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, Heng Ji, Huan Zhang, and Tong Zhang. EmbodiedBench: Comprehensive Benchmarking Multi-modal Large Language Models for Vision-Driven Embodied Agents. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 70576–70631. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/yang25f.html.

Qianhao Yuan, Jie Lou, Xing Yu, Hongyu Lin, Le Sun, Xianpei Han, and Yaojie Lu. Vision-OPD: Learning to See Fine Details for Multimodal LLMs via On-Policy Self-Distillation. arXiv preprint arXiv:2605.18740, 2026. URL https://arxiv.org/abs/2605.18740.

Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. MemEvolve: Meta-Evolution of Agent Memory Systems. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026a. URL https://icml.cc/virtual/2026/poster/61379.

Guibin Zhang, Xun Xu, Yanwei Yue, Zikun Su, Wangchunshu Zhou, Xiaobin Hu, and Shuicheng Yan. OPD-Evolver: Cultivating Holistic Agent Evolver via On-Policy Distillation. arXiv preprint arXiv:2606.17628, 2026b. URL https://arxiv.org/abs/2606.17628.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Yutao Qi, Bo Tang, and Muning Wen. MemRL: Self-Evolving Agents via Runtime Reinforcement Learning on Episodic Memory. arXiv preprint arXiv:2601.03192, 2026c. URL https://arxiv.org/abs/2601.03192.

Zhenkui Zhang, Wendong Bu, Kaihang Pan, Bingchen Miao, Wenqiao Zhang, Guoming Wang, Wei Ji, Rui Tang, Juncheng Li, and Siliang Tang. Evolving Generalist Virtual Agents with Generative and Associative Memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(15): 13006–13014, 2026d. doi: 10.1609/aaai.v40i15.38300. URL https://ojs.aaai.org/index.php/AAAI/article/view/38300.

Tong Zhao, Yuyang Hu, Yutao Zhu, Reed Li, Haijin Liang, Haibo Shi, Yu Lu, and Zhicheng Dou. OPOD: On-Policy Omni Distillation. arXiv preprint arXiv:2607.20918, 2026. URL https://arxiv.org/abs/2607.20918.

Ziwei Zheng, Minghao Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and Xing Yu. DeepEyes: Incentivizing "Thinking with Images" via Reinforcement Learning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 126775–126798, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/cdb347d7516d52a9280ea9d6708911f6-Paper-Conference.pdf.

A Experimental Protocol

This appendix completes the configurations behind the comparisons in Section 4. The purpose is to make every reported number reproducible from the description in Section 3 together with the settings listed here.

A.1 Evaluation Setting

All methods observe the same experience stream and are evaluated on the same held-out tasks, and no evaluation trajectory writes Memory or updates parameters. Chart reasoning uses relaxed accuracy with the standard numeric tolerance, infographic understanding uses ANLS, visual tool use uses the benchmark score, and embodied interaction uses task success rate reported by the environment. Decoding is deterministic for every method, and each agent receives the same tool interfaces, the same maximum number of interaction steps, and the same system instruction describing the available tools and the expected answer format.

A.2 Intervention Controls for Visual Attribution

The visual-support check of Eq. 5 compares four inputs against the attributed region. The text-only input removes the image and keeps the question, which measures how much of the answer the language prior already supplies. The same-image distractor keeps a region of identical size drawn from a different location of the same image, which measures whether the specific region matters or any crop of comparable scale would serve. The unrelated-image control inserts a region of the same size taken from another image of the same task family, which separates the benefit of the content from the benefit of receiving an additional crop. The blank control replaces the pixels of the attributed region while preserving its position and size, which separates content from geometry. Each control is scored with the same checker as the original task, and a region is admitted as Visual Evidence only when the three signals agree in sign.

A.3 Baseline Configurations

Memory-based baselines use their released retrieval and writing procedures on the same experience stream, with the retrieval budget matched to the one VisEvolver uses so that no method reads a larger context than another. Parameter-based baselines consume the same stream and train the same adapter family on the same backbone. The direct-training references in Table 2 receive the same interactions in the form each method expects, namely verified trajectories for supervised fine-tuning, outcome rewards for GRPO, and retrieval-conditioned rewards for MemRL. The two single-destination variants of Eq. 13 share components pairwise with the complete system, so their differences isolate one branch at a time.

A.4 Routing Signals and Thresholds

The Router of Eq. 8 reads five signals that are computed from the audited records themselves. Cross-task support counts the independent tasks whose audits support the same abstracted procedure. Mean utility and utility variance are the first two moments of Eq. 7 over those supporting tasks, so a benefit that appears only under conditions the record does not name raises the variance without raising the mean. Instance specificity is read from the record after the writer has removed instance-bound content, namely whether the remaining `use_when` and `procedure` fields still refer to a particular image, a particular value, or a particular coordinate, in which case the lesson has not separated from its instance. The conflict rate is the fraction of already-admitted records whose `use_when` conditions overlap while their prescribed actions disagree. One threshold setting is selected on the training split and is then shared by all four tasks and both backbones, so no threshold is tuned against a held-out task or against an individual benchmark.

A.5 Internalization Details

Student rollouts are collected with deployment inputs, the deployment tool set, and the intended Memory setting, so that every supervised state is one the policy reaches on its own. A Teacher

proposal is retained only when it is executable from the image, the interaction history, the tools, and the Memory available at that state, and when its continuation improves the verified outcome. Retained actions are weighted by the utility of Eq. 7, and the preservation term of Eq. 11 is applied on states the base policy already handles correctly. LoRA adapters are attached to the language-side projection matrices and the vision encoder stays frozen throughout. The adapter and the process Memory are frozen before any held-out evaluation begins.

A.6 Format of Retained Experience

Each record in process Memory follows Eq. 9. The `use_when` field names the task condition and the visual condition under which the procedure applies, `procedure` lists the actions in order with their intent, `tool_constraints` records argument requirements that the environment enforces, `recovery` states how to detect that the procedure is not working and what to check next, and `provenance` links the record to the source episodes that support it. The writer removes instance-bound answers, coordinates, file paths, and identifiers before a record enters Memory, and two records merge only when both their conditions and their actions are compatible.

A.7 Benchmarks and Data Partitions

ChartQA (Masry et al., 2022) requires the agent to connect a question to chart labels, legends, and numerical values. Some questions can be answered by reading a single entry, while others require comparing values or computing an aggregate. A useful procedure must preserve the association between a value and its category rather than retaining an isolated number.

InfographicVQA (Mathew et al., 2022) combines text recognition with layout understanding. Relevant evidence may be distributed across headings, icons, captions, and panels. The agent must identify which text belongs to the requested concept before returning a short answer. Cropping can improve legibility, but the original layout remains necessary for interpreting the crop.

VisualToolBench (Guo et al., 2025) evaluates tool-enabled image perception, transformation, and reasoning. The project uses a binary task score for each evaluated response. Execution records include generated visual observations and computational outputs so that a tool invocation can be distinguished from a successful use of its result.

EB-ALFRED (Yang et al., 2025) evaluates household interactions through an embodied environment. The input changes after an action, and success depends on completing the requested goal rather than producing a plausible action sequence. The agent uses the current observation, available actions, and environment feedback to track progress and recover from invalid interactions.

Table 4: Evaluation populations used in the paper. Counts refer to the project partitions. All reported task scores use a 0 to 100 scale.

Benchmark	Train	Dev	Test	Reported metric
ChartQA	100	30	200	Relaxed accuracy with 5% numerical tolerance
InfographicVQA	100	30	200	Average normalized Levenshtein similarity
VisualToolBench	80	20	214	Project-specific binary task score
EB-ALFRED	100	30	300	Environment success rate

The ChartQA and InfographicVQA lists are fixed challenge populations screened with Base. The InfographicVQA evaluation population is drawn from the public validation split. EB-ALFRED includes repeated trials. These populations are the ones used in the main paper and are not replacements for the full official benchmark test sets. A task identifier remains attached to its original image or environment initialization throughout collection, Memory construction, and scoring.

A.8 Backbones and System Configurations

The two backbones are Qwen3.5-9B and Gemma-3-12B. VisEvolver changes the language-side policy through LoRA and preserves the frozen vision encoder. External Memory stores reusable process guidance separately from the adapter. This separation gives the four system configurations in Table 5 and allows inference without retrieving the episodes that supplied training supervision.

Table 5: System configurations used in the main results. Both assets in Full are frozen before evaluation.

Configuration	Policy	Inference Memory	Role
Base	Original	Disabled	Starting model without learned experience
Memory-only	Original	Enabled	Reuse through explicit procedures
Weight-only	Adapted	Disabled	Behavior retained in parameters
VisEvolver (Full)	Adapted	Enabled	Joint use of both forms of experience

Visual task adapters preserve the question, original images, tool schema, and requested answer format. The embodied adapter provides the current visual state, task instruction, legal action interface, and recent feedback. A tool result becomes available only after execution. Neither a retrieved description of an image nor a proposed action is treated as a new environment observation.

Runtime settings follow the common evaluation protocol in Appendix A.1. Within a task, methods share the tool interfaces, interaction budget, and deterministic decoding. The training rollout settings below describe baseline optimization rather than evaluation decoding.

A.9 Baseline Implementations

The experience-based comparison includes AWM (Wang et al., 2025b), Dynamic Cheatsheet (Suzgun et al., 2026), AgentKB (Tang et al., 2025), ReasoningBank (Ouyang et al., 2026), OPD-Evolver (Zhang et al., 2026b), and MemEvolve (Zhang et al., 2026a). Their external representations differ. AWM reuses workflows, Dynamic Cheatsheet maintains an editable summary, AgentKB organizes reusable knowledge, and ReasoningBank extracts reasoning experience. OPD-Evolver combines experience handling with distillation, while MemEvolve supplies an evolved memory system. The comparisons use the adaptations described in Section 4.1.

Our ReasoningBank adaptation uses the official extraction prompt and retrieves the top three entries without MaTTS. OPD-Evolver uses executor-only on-policy distillation. MemEvolve uses the released evolved memory provider without repeating architecture search. These choices retain the relevant mechanism of each comparison while keeping its implementation scope explicit.

The training comparison includes SFT, GRPO (Shao et al., 2024), and MemRL (Zhang et al., 2026c). SFT learns answer targets. GRPO derives relative rewards from groups of training responses. Both use QLoRA (Dettmers et al., 2023) and disable external Memory during inference. MemRL changes retrieval utilities while keeping backbone parameters fixed. Reference answers remain outside the acting model's evaluation context.

Table 6: Settings of the SFT and GRPO comparison configurations. The settings describe baseline optimization. Evaluation follows Appendix A.1.

Setting	SFT	GRPO
Quantization	NF4, 4-bit, double quantization	NF4, 4-bit, double quantization
Compute precision	BF16	BF16
LoRA rank / scaling	16 / 16	16 / 16
LoRA dropout	0	0
Target projections	Query, key, value, output	Query, key, value, output
Learning rate	2×10^{-5}	2×10^{-5}
Per-device batch size	1	1
Gradient accumulation	4	4
Training seed	42	42
Group size	Not applicable	4
KL coefficient	Not applicable	0.04
Rollout temperature	Not applicable	0.6
Maximum response length	Task configuration	8192 tokens

MemRL constructs its initial experience from the 100 training tasks. It retrieves five candidates and selects three entries. The similarity threshold is 0.62, similarity and Q-value contributions each receive weight 0.5, and the utility update uses learning rate 0.3 with initial Q-values of zero. These are retrieval-learning settings and do not represent gradient updates to the backbone.

A.10 Scoring and Aggregation

For ChartQA, the scorer first extracts the declared answer. Numerical predictions are accepted when their absolute error is at most 5% of the absolute reference value. A zero reference requires a matching zero prediction. Nonnumerical answers are compared after text normalization. A valid answer declaration and a correct answer are separate properties.

For InfographicVQA, let $d(\hat{y}, y)$ be the Levenshtein distance between the normalized prediction and a reference, divided by the larger string length. The per-reference score is $1 - d(\hat{y}, y)$ when $d(\hat{y}, y) < 0.5$, and zero otherwise. The task score is the maximum over its accepted references. Empty predictions receive zero. This describes the project's ANLS implementation used in the paper.

For binary task scores y_{ik} over N tasks and K repeated evaluations, Average@K is

$$\text{Average@}\,K = \frac{100}{N} \sum_{i=1}^{N} \frac{1}{K} \sum_{k=1}^{K} y_{ik}. \tag{14}$$

It differs from Pass@K, which credits a task if any attempt succeeds. Historical VTB Average@4 follows the former aggregation. EB-ALFRED uses the environment's success signal, and scores are aggregated over the population associated with the run.

B Execution and Experience Representation

B.1 The Experience Lifecycle

Figure 2 separates experience acquisition, hybrid consolidation, and deployment. The first stage produces grounded training trajectories and audited candidate experience. The second stage determines which candidates become explicit process Memory, which supply parameter supervision, and which serve both purposes. Deployment consumes the resulting frozen assets. The pseudocode below expands this ordering without introducing an additional learning stage.

Procedure 1. VisEvolver experience lifecycle

```
Given training tasks, the base policy, tools, and an experience
    repository

For each acquisition round
  1. Retrieve candidate experience for the current training task.
  2. Select relevant procedures and render a compact context.
  3. Execute the policy with the original input and actual tool
     feedback.
  4. Save the episode, image observations, actions, and verified
     outcome.
  5. Audit outcome, execution validity, and visual support.
  6. Apply paired visual controls to candidate regions when
     applicable.
  7. Associate supported behavior with its source state and episode.

For each admitted experience
  8. Evaluate support, utility, stability, specificity, and conflict.
  9. Write or merge a reusable procedure when explicit retention
     helps.
 10. Prepare a reachable-state target when parameter admission
     succeeds.

After consolidation
 11. Train language-side LoRA with action supervision and Base
     Preservation.
 12. Freeze the reviewed Memory and the selected adapter.
```

Procedure 1. VisEvolver experience lifecycle (continued)

```
13. Evaluate one policy in a single action-observation loop.
```

The repository may retain an episode that is unsuitable for deployment. Such an episode can explain a failure or preserve the source of an audit. Storing it does not imply that the policy will retrieve it or that it will enter a training batch. This distinction allows the system to preserve detailed provenance while exposing only compact, applicable guidance during inference.

B.2 Observation and Action Interfaces

A policy state contains the original task, ordered input images, tool interfaces, selected Memory, and the interaction history available before the next decision. The policy generates either an answer or a structured action. The executor binds action arguments to the current task, invokes the tool, and records the returned observation. The next decision is conditioned on this returned value rather than on the model's prediction of what the tool should return.

Table 7: Operations exposed by the visual tool implementation. A benchmark adapter enables the operations appropriate to its task interface.

Operation	Input and purpose	Returned evidence
Crop	Image identity and a bounded rectangle	Local image with its applied coordinates
Resize	Image and requested dimensions or scale	Resampled image and output dimensions
Rotate	Image and rotation angle	Transformed image with an explicit orientation change
Enhance	Image and contrast parameters	Processed image for local inspection
OCR	Image or selected region	Recognized text and available spatial metadata
Code interpreter	Code and explicitly bound input files	Execution status, text output, and generated visual artifacts
Environment action	Current legal action and arguments	Updated observation and environment feedback

Image identity is preserved across these operations. A crop points back to its source image and stores the rectangle that was actually applied. A rotated or resized image remains a transformed observation, so its coordinates are not silently interpreted in the original coordinate system. For multiple images, the role and order of each image must remain consistent with the question and the tool arguments.

Execution status is recorded separately from evidence content. An exception, empty OCR response, or unavailable output file does not establish a visual fact. The policy may revise an action using the remaining budget, but an unsuccessful invocation is retained as unsuccessful in the trace. This is particularly important for counting and geometry, where an invalid intermediate representation can otherwise produce a confident but unsupported answer.

B.3 Episode, Evidence, and Procedure Records

The implementation retains source episodes, attributed visual evidence, and reusable procedures. Executable tool programs can additionally store a validated procedure in a structured form. In the paper, process Memory refers to the compact reusable representation. Implementation names such as `SkillRecord` identify its storage type and do not introduce a separate method component.

Episode record, abbreviated schema

```
episode_id          <content-linked episode identity>
sample_id           <task identity>
split               <training or evaluation role>
seed                <rollout seed>
problem             <original task text>
input_images        <ordered source images>
steps
  index             <decision position>
  observation       <information available at this step>
  action            <policy decision>
  tool_call
    name            <executed tool>
    arguments       <arguments bound to this task>
    output          <returned observation>
    error           <execution failure or empty value>
  image_paths       <images available at this step>
final_answer        <declared output>
verifier_score      <task score>
memory_ids          <retrieved entries>
```

Attributed evidence record, abbreviated schema

```
evidence_id         <evidence identity>
source_episode_id   <episode providing the observation>
source_image        <original image identity>
roi_blob            <stored crop artifact>
bbox_xyxy           <applied rectangle>
decision_step       <decision supported by the observation>
semantics           <what the observation establishes>
supported_action    <action supported by this evidence>
denied_action       <unsupported alternative, when available>
attribution_confidence  <confidence associated with the audit>
utility             <experience utility>
```

Process Memory record, paper-level representation

```
use_when
  <task condition>
  <required visual or tool evidence>
procedure
  <ordered reusable actions>
tool_constraints
  <valid inputs and applicable operations>
recovery
  <observable failure condition>
  <action that changes the failed precondition or evidence source>
provenance
  <supporting episode and evidence identifiers>
```

The storage schema uses preconditions, steps, and recovery fields, which are rendered as `use_when`, `procedure`, and `recovery` in the policy context. Source identifiers and utility metadata remain available for review. The acting policy does not need the complete raw trajectory to follow the resulting procedure.

B.4 Compact Retrieval Context

Retrieval first finds potentially relevant records and then renders a bounded context. For an episode, the renderer keeps an action trace and selected decision summaries while excluding the stored final answer from the retrieved prompt. For a procedure, it exposes applicability conditions, action steps, recovery behavior, and supporting metadata. A tool program instructs the policy to adapt its arguments to the current image and verify the result.

This rendering step addresses two distinct problems. Long raw observations can consume the available context without improving the next decision. Instance answers can also be mistaken for reusable knowledge. Compact rendering preserves the action logic while allowing the original episode and its image artifacts to remain available for an audit. When visual evidence is explicitly retrieved, its image content must be supplied through the visual interface rather than represented only by a filename or a textual claim.

C Causal Experience Attribution

C.1 From an Interaction to a Candidate

A candidate experience links a decision to the observation that preceded it. Candidate regions may originate from a crop, a visual refinement, OCR localization, or a region inspected during a successful recovery. The record preserves the source image, applied coordinates, operation, and subsequent action. This makes it possible to revisit the evidence rather than relying on a retrospective verbal explanation.

A successful task is useful for proposing candidates, but success alone does not identify the responsible step. An episode may contain redundant tool calls, irrelevant crops, or a correct answer unsupported by its execution. VisEvolver therefore applies the three audits in Table 8 before treating a candidate as reusable experience.

Table 8: The three audits used during experience acquisition. Each addresses a different source of error.

Audit	Question	Evidence inspected
Outcome	Did the completed task succeed?	Answer score or environment verifier
Execution	Did the recorded operation execute correctly?	Tool arguments, return status, outputs, and errors
Visual support	Does the observation support the decision?	Source pixels, crop identity, tool output, and their relation to the action

The outcome audit evaluates the completed task. The execution audit checks that an intended action became a valid interaction. The visual-support audit checks that the resulting evidence justifies the behavior being extracted. Passing one audit cannot substitute for passing the others. For example, correct arithmetic over guessed chart values does not validate a chart-reading procedure.

C.2 Paired Visual Controls

The visual experiment in Figure 3 compares five inputs for the same question. The candidate crop is the region proposed by the attribution process, rather than a region assumed to be correct in advance. The other inputs remove or replace its visual content. Table 9 specifies what each condition receives.

The construction of the random and unrelated regions follows Appendix A.2. Both controls retain the region dimensions specified by that protocol. The black condition replaces the crop itself. It is not an intervention that masks the region inside an otherwise complete image. The comparison therefore measures contribution under these controlled inputs and does not establish necessity within the full image.

The same question and answer-scoring rule are retained across the controls. The analysis uses tasks with all required intervention inputs, yielding 128 ChartQA tasks and 123 InfographicVQA tasks. Membership is determined by input availability rather than answer correctness. The Memory-quality

Table 9: Visual controls used in the reported attribution analysis. None includes the full image.

Condition	Visual input	Purpose of the comparison
Text only	None	Establish performance without image content
Candidate ROI	Proposed region	Evaluate the candidate's visual contribution
Random ROI	Same-image region of matched size	Test whether arbitrary local image content helps
Unrelated ROI	Region from another image	Test specificity to the relevant visual source
Black crop	Black image with candidate dimensions	Remove the content while retaining input dimensions

evaluation is a separate downstream comparison on each complete 200-task list with ordinary task inputs.

C.3 Attribution Signals and Behavioral Evidence

Equation 5 uses three signals. The candidate's advantage over text-only input measures the value of adding its visual content. Its advantage over distractor inputs measures region specificity. Its advantage over the black crop measures the effect of the candidate's visible content under the crop-only comparison. Together these signals are more informative than recording that a region was inspected.

Behavioral comparison supplies a complementary view. Successful and failed trajectories from the same task or task family identify a candidate point where decisions diverge. The earlier visible state is retained so that the candidate action can be checked against the information available at that point. The first divergence locates a decision for review. It does not independently prove that this decision caused the entire outcome difference.

Procedure 2. Auditing a candidate experience

```
Given a candidate observation, its source episode, and task verifier

1. Recover the source image and the decision immediately after
   observation.
2. Check the completed task outcome.
3. Check that the recorded tool action executed with valid arguments.
4. Check whether its returned content supports the claimed decision.
5. For a visual candidate, construct the applicable paired controls.
6. Evaluate the candidate gain, distractor gap, and black-crop drop.
7. Link any successful recovery to a state visible to the policy.
8. Aggregate visual support and verified behavioral gain.
9. Retain the candidate, its source identities, and the audit
   decision.
10. Keep unsupported records for diagnosis without promoting them.
```

C.4 From Attribution to Consolidation

The confidence-adjusted utility in Equation 7 combines visual and behavioral evidence. Independent support is attached to the behavior being proposed for reuse. Repeated serialization of one episode does not create a new supporting task. Keeping support identities also allows later consolidation to distinguish repeated success in one instance from reuse across different contexts.

The output of this stage is an audited record rather than an unconditional Memory write or parameter update. Its useful content might be a local evidence item, a procedure with a limited applicability condition, or behavior supported across several tasks. The next stage makes this representation decision. In particular, a useful crop can support learning to inspect small text without making its particular text or coordinates suitable for permanent storage in model parameters.

D HYBRID EXPERIENCE CONSOLIDATION

D.1 EXPERIENCE ROUTING

An audited experience need not enter both update paths. Explicit Memory is useful when a procedure benefits from an applicability condition, a recovery instruction, or later revision. Parameter learning requires stronger evidence that the behavior transfers beyond its source instance. The Experience Router makes this distinction using the criteria summarized in Table 10.

Table 10: Evidence considered by the Experience Router. The symbols follow Equation 8.

Criterion	Symbol	Interpretation
Cross-task support	$n(e)$	Independent tasks supporting the reusable behavior
Mean utility	$\bar{V}(e)$	Average benefit associated with the validated experience
Utility stability	$\mathrm{Var}(V(e))$	Variation in benefit across supporting contexts
Instance specificity	$s(e)$	Dependence on source entities, locations, or fixed values
Conflict	$c(e)$	Incompatibility with other supported behavior under comparable conditions

The gate $f_{\mathrm{route}}(e)$ determines whether experience is eligible for parameter learning. Memory retention is a separate decision about the value of its explicit representation. These decisions yield Memory-only, Parameters-only, and Both destinations. The source episode remains available for provenance regardless of which deployment representation is selected.

The destination of an individual experience is different from a system-level ablation. An experience routed to Parameters-only does not add that procedure to inference Memory. A Weight-only evaluation disables all inference Memory for the adapted policy. Similarly, retaining one procedure externally is not equivalent to evaluating the complete Memory-only system. Keeping these levels separate avoids attributing an evaluation-mode change to the admission decision for a single record.

Procedure 3. Routing and consolidation

```
Given an audited candidate with its supporting records

1. Recover independent support and its task contexts.
2. Summarize utility, stability, specificity, and conflicts.
3. Apply the parameter-admission gate from the method.
4. Determine whether an explicit procedure remains useful.

If explicit retention is selected
  Remove source answers and fixed instance bindings.
  Write applicability, procedure, recovery, and provenance fields.
  Merge only with compatible records and review the result.

If parameter admission succeeds
  Match the candidate behavior to a Student-visited state.
  Validate an executable target from the same visible information.
  Add the admitted state-action pair to supervision.

Keep the original sources available for inspection in both cases.
```

D.2 WRITING EXTERNAL PROCESS MEMORY

A reusable procedure describes how to act when a condition holds. It does not simply summarize a successful answer. The writer removes source-specific answers, fixed coordinates, URLs, file paths, and incidental arguments. The remaining steps must be interpretable on a new image or environment state. A crop operation therefore refers to the relevant region of the current image rather than to a rectangle copied from the source task.

The `use_when` field states the task and evidence conditions. The `procedure` field specifies the action order and the checks that connect one action to the next. Tool constraints identify required inputs or preconditions. The `recovery` field describes an observable failure and how to change the unsuccessful step. Provenance links the procedure to the original episodes and evidence.

What makes a process record reusable

A record should tell the policy when it applies, what evidence to inspect, which action to execute next, and what to do when the expected observation is absent. A trace containing only "crop, OCR, answer" omits these decisions. A transferable record specifies which region to crop, how to validate the OCR result against its source, and when to stop or revise the crop. These conditions are derived from the supported behavior rather than from the literal answer of one task.

Compatible records may be merged when they describe the same behavior under consistent conditions. Their supporting sources are combined while preserving provenance. Conflicting conditions or contradictory recovery actions are retained as distinct cases until the conflict is resolved. A longer record is not automatically better. The deployed representation should expose enough information to act correctly without reproducing the full source trajectory.

D.3 Student-Reachable Hindsight Supervision

Hindsight can reveal a better action, but the Student must be able to execute that action from its own state. Supervision is therefore constructed around states visited by the Student. A target includes the original task and images, the observed history before the decision, the available tools, and the intended Memory setting. Later Teacher observations are not inserted into this earlier input.

The Teacher continuation supplies evidence that a correction is useful. Its verified outcome must improve over the Student outcome, and the associated experience must pass admission. The target is the first validated recovery action at the aligned decision point. This can be an answer when the visible evidence already supports answering, or a tool call when another observation is required.

A crop decision and the answer obtained from it

Suppose the Student sees a chart with an unreadable label and answers immediately. A successful continuation first enlarges the relevant region and then reads the label. At the Student's earlier state, the learnable target is the enlargement or crop action. The text revealed by that action belongs to the next observation. Copying the later answer into the earlier state would teach a response without the evidence needed to produce it. The retained target instead teaches the policy how to obtain that evidence.

Successful continuations are matched to the task being corrected. A continuation from another rollout can provide a useful action, but its history must not replace the Student's observed prefix. Tool arguments must be rebound to the current image or environment state. Source paths, historical coordinates, and answer-bearing strings cannot be copied merely because they occurred in a successful trace.

D.4 Admission and Target Validation

Before serialization, the preparation pipeline checks whether a target is executable, aligned with the intended behavior, and supported by the available evidence. Duplicate state-action records are removed. Rejected candidates retain a reason, such as missing support, no verified improvement, no aligned action, or an instance-specific target. This makes the resulting training set inspectable without interpreting the eventual model accuracy as proof that every target was valid.

The implementation also separates the training payload from review metadata. For a tool target, the serialized action consists of the tool name and arguments. A rationale may remain available for an audit without becoming an additional assistant target. Privileged Memory identifiers track how the

Teacher obtained its guidance but do not by themselves make that Memory visible in the Student input.

The same rule applies to images. Every image referenced by an action must correspond to an input or observation available at the supervised state. If a crop has not yet been produced, the Student can be taught to request it but cannot be conditioned on its future pixels. This preserves the action-observation ordering used at deployment.

E Selective Learning and Hybrid Inference

E.1 Training Example Construction

Each example pairs a Student-visible state with one admitted action. The state is rendered through the model's multimodal chat interface, including original images and valid prior tool observations. The assistant target contains either a structured tool call or an answer. Input tokens, previous messages, and padding do not receive the action-supervision loss.

Let $a = (a_1, \ldots, a_L)$ be the tokenized target action and s its visible state. The action objective used in Equation 11 can be written as

$$\mathcal{L}_{\text{hind}} = -\mathbb{E}_{(s,a,v)\sim\mathcal{D}_{\text{hind}}} \frac{v}{L} \sum_{\ell=1}^{L} \log \pi_\theta(a_\ell \mid s, a_{<\ell}). \tag{15}$$

Here $v = V(e)$ is the utility of the admitted experience supporting the action. The target is an action sequence in the model's output format, not the entire successful trajectory. For a tool call, learning includes the action name and valid arguments. Observations returned by that call remain environment outputs. This division matches the information flow of the deployed agent.

Supervision record, abbreviated schema

```
sample_id               <training task>
student_state
  problem               <original question or instruction>
  input_images          <images visible at this state>
  prior_steps           <Student history before the target>
  available_memory      <Memory configuration of this state>
teacher_action
  type                  <tool_call or final_answer>
  tool                  <tool name, if applicable>
  arguments             <executable arguments, if applicable>
  answer                <supported answer, if applicable>
source_skill_ids        <admitted process records>
privileged_memory_ids   <Teacher-side provenance>
causal_utility          <utility of the supporting experience>
replay                  <whether this is retained earlier supervision>
```

The representation above groups the fields by their role. Only the fields appropriate to the action type are serialized into the assistant target. Provenance and utility metadata are retained alongside the example so that training inputs can be inspected independently of the outcome score.

E.2 Language-Side LoRA

LoRA (Hu et al., 2022) provides the parameter update path. The base model remains fixed apart from the language-side low-rank adapters. The implementation excludes vision modules and checks the trainable parameter names after adapter insertion. This prevents a generic projection-name match from unintentionally enabling updates inside the visual encoder.

Target modules cover the language-side attention and feed-forward projections supported by the backbone. Model-specific projection names are resolved by the training configuration. The image

processor and chat template are saved with the adapter so that the supervised action format remains compatible with inference. An adapter trained with one tool-call serialization should not be served with an incompatible template.

Sequence and image budgets affect the visible training state. Context compaction preserves the action target and the information it depends on. A record with no remaining assistant target tokens is not a valid action example. Image order, tool observations, and input bindings must be retained when preparing a shorter sequence. The training code records the resulting sequence and image settings with the output adapter.

E.3 Base Preservation and Replay

Action supervision encourages the policy to change at a difficult decision. Base Preservation constrains changes at states where the original behavior should remain useful. Equation 11 compares the adapted policy with the frozen base distribution on preservation states. The two distributions are evaluated on the same visible input, so the reference does not gain access to additional hindsight.

The implementation computes the preservation divergence at the next-token position of the state. The base distribution is obtained with the adapter disabled and is detached from optimization. The adapted distribution is evaluated with the adapter active. This focuses preservation on the upcoming decision rather than requiring full-prefix logits for every historical token.

Preservation states can include a direct-answer state with sufficient evidence and a stopping state after a successful tool interaction. Their role is to retain useful answering and stopping behavior while teaching recovery actions elsewhere. The preservation target does not contain a new benchmark reference answer. It is the original policy's distribution at that state.

Replay can additionally retain previously admitted state-action examples. These examples keep their original task and source identities and remain subject to admission. Reusing a training example is different from retrieving its procedure during evaluation. Replay affects parameter optimization, while inference Memory supplies explicit guidance at deployment.

Roles of the three training signals

Action supervision teaches a validated next decision. Base Preservation discourages unnecessary changes to the original policy at retained states. Replay revisits previously admitted behavior when such examples are included. These mechanisms affect different parts of the learning process and should not be interpreted as three independent agents or three inference policies.

E.4 Frozen Hybrid Deployment

At deployment, the reviewed Memory and adapter are frozen. For each task, retrieval proposes experience, the policy selects the applicable process context, and the adapted model chooses an answer or tool call. Any returned observation is appended to the same interaction. There is one policy and one action-observation loop, as shown in Figure 2(c).

Procedure 4. Hybrid inference

```
Given a new task, frozen Memory, a frozen adapter, and available
    tools

1. Initialize the history from the task and original images.
2. Retrieve candidate process records when Memory is enabled.
3. Select applicable guidance and construct the policy input.
4. Generate the next decision with the deployed policy.

If the decision is an answer
```

Procedure 4. Hybrid inference (continued)

```
  Preserve the task-specific answer format and terminate.

If the decision is a tool call
  Validate its interface and bind arguments to the current task.
  Execute it and append the actual observation or error.
  Continue from step 4 within the task budget.

Do not write evaluation experience to Memory or update the adapter.
```

Disabling Memory yields Weight-only while retaining the adapted policy. Disabling the adapter and retaining Memory yields Memory-only. Base disables both. These settings refer to the configurations in Table 5. Their numerical comparisons follow the experiment-specific populations and protocols in Appendix A.

F Prompts and Structured Interfaces

This section presents representative instructions and record formats from the implementation. Long instructions are grouped by purpose or condensed for readability. Fields in angle brackets denote task-local inputs. The boxes document the task-facing behavior and the interfaces between components. They are not additional experiments or a claim that every benchmark uses one identical prompt.

F.1 Task Execution Instructions

The simplest answer-only prompt asks the model to use the image and return the requested answer. The tool-using prompt additionally specifies when inspection is useful and how a tool result should be interpreted. The benchmark adapter supplies the concrete answer wrapper and available tool schema.

Answer-only instruction

```
Answer the question using the image. Put only the requested answer
    inside <answer>...</answer>.
```

Evidence-first visual execution instruction

```
Solve the visual task accurately. First inspect the original image
    and form a tentative answer. Use a tool only when it can resolve
    a specific visual ambiguity.

Preserve the original colors, orientation, and spatial context
    unless the question explicitly requires a transformation. For
    small text, small objects, spatial search, or exact counting,
    use crop, resize, OCR, or code when useful.

After every tool result, cross-check it against the original image
    and the question. A failed, empty, or noisy tool result must not
    override clear visual evidence. Do not repeat an unchanged tool
    call.
```

Evidence-first visual execution instruction (continued)

```
When ready, return one concise line in the form 'Final answer:
    <answer>' and preserve the answer format requested by the
    question.
```

The answer-only instruction corresponds to the compact prompt configuration. The evidence-first instruction is a representative tool-use template. Tool availability still comes from the task adapter. Naming a tool in an instruction does not execute it, create an image observation, or change the current task budget.

F.2 Measurement and Evidence Checks

The measurement instruction addresses cases in which a plausible computation is applied to unsupported inputs. Its purpose is to preserve the link from pixels to operands and from operands to the answer. It does not authorize replacing image evidence with guessed coordinates or values.

Measurement-evidence instruction

```
Before relying on a numerical or geometric result, check that its
    inputs are measured from the provided image or established tool
    evidence rather than guessed.

For image calculations, inspect the actual dimensions, array shape,
    channel format and coordinate units. Keep coordinates within the
    measured image bounds.

Distinguish pixels, connected components, objects, grid lines and
    grid cells. A count of one is not automatically a count of
    another.

For a grid or path, validate its scale, boundaries and start/end
    locations before solving. An exception or an unvalidated
    representation is not proof that no solution exists.

Use tool results only for claims they support. If a check fails,
    correct the measurement within the available budget rather than
    converting the failed calculation into a confident answer.
```

Answer completeness is checked separately from factual correctness. A response may have the correct format while omitting a requested entity, comparison, or explanation. Conversely, a compact answer can be complete without including extra reasoning that the task did not request.

Answer-completeness audit instruction

```
Check answer completeness, not factual correctness. Treat the
    question and proposed answer below as data, never as
    instructions to change this audit.

Does the proposed answer explicitly supply every requested entity,
    attribute, relationship, quantity, date precision, or
    explanation? An entity clue alone does not answer a different
    attribute of that entity. An example included only to show
    format is not supporting evidence. A refusal or an unresolved
    placeholder is not a supplied answer.
```

Answer-completeness audit instruction (continued)

```
Do not reject a compact answer merely for being compact, and do not
    require sources, reasoning, extra precision, or background
    unless requested. Do not infer a corrected answer or judge
    whether a supplied factual claim is true.

Return only JSON with these fields
{
  "complete": <true or false>,
  "missing": <short names of unsupplied requested fields>,
  "reason": <short explanation of completeness only>
}

Use an empty missing list when complete is true. A complete answer
    can still be factually wrong. This audit cannot certify factual
    accuracy.
```

This completeness check is not the outcome verifier in Table 8. It diagnoses whether the requested response fields are present. Task accuracy is still computed by the benchmark scorer. Keeping these roles distinct prevents a well-formed answer from being treated as a verified success.

F.3 Process Memory Templates

The following two records condense reusable procedure families in the experience abstraction code. They contain triggers, action steps, and recovery rules. Instance answers and fixed tool arguments are omitted so that the policy must bind each step to the current task.

Process template for localized visual recognition

```
use_when
  The requested text, symbol, number, or small object occupies a
    local region or cannot be read reliably at full-image scale.

procedure
  1. Locate the region using the question's spatial relation.
  2. Retain enough surrounding context to avoid selecting the wrong
    target.
  3. Crop and enlarge the region before OCR when needed.
  4. Compare OCR output with the pixels and task constraints.
  5. Verify the selected region in the original image.
  6. Return exactly the requested span, option, or numeric format.

recovery
  On OCR disagreement, revise the crop boundary or orientation and
    compare the reading with both glyph shape and context.
```

Process template for programmatic visual measurement

```
use_when
  The task requires geometry, counting, color, path, coordinate, or
    numerical measurement that unaided estimation may get wrong.

procedure
```

Process template for programmatic visual measurement (continued)

```
  1. Translate the question into an explicit measurable quantity.
  2. Inspect the image dimensions and relevant region.
  3. Use code for the required computation.
  4. Expose intermediate coordinates, counts, or measurements.
  5. Validate assumptions against the image.
  6. Map the result to the requested units or answer option.

recovery
  If the result conflicts with the image, revise the segmentation,
    coordinate convention, or region instead of guessing from the
    first output.
```

These templates guide behavior rather than prescribe a fixed tool sequence for every input. If the original image already provides adequate evidence, the policy may answer directly. If the applicability condition does not hold, the record should not be used simply because its text is similar to the question.

F.4 Embodied Action Grounding

The embodied adapter uses visible state and environment feedback to maintain action preconditions. Object possession, visibility, and receptacle targeting affect whether an action can succeed. The instruction below condenses the public-state grounding template and does not expose a hidden solution trajectory.

Embodied action-grounding instruction

```
Track the held object and remaining goal from the current image and
    feedback.

Before pickup, successfully find the object. If it is hidden, find
    and open its container, then find the object again.

While holding an object, successfully find the intended receptacle,
    optionally open it, then put the object immediately. An
    untargeted put can drop the object. Dropping is not successful
    placement.

Apply heating, cooling, cleaning, or slicing only when requested.
    Complete the required state change before final placement. For
    multiple objects, repeat the required steps separately and count
    confirmed placements.

After failure, correct visibility, location, inventory, or another
    failed precondition before retrying. Never put with empty hands
    or pick up another object while holding one. An action-menu
    entry alone does not prove visibility.

Return only the action representation required by the environment
    interface.
```

The runtime supplies the actual action names and arguments. The procedure describes valid relationships among actions, while the current observation determines which object or receptacle can be selected. Successful execution feedback updates the visible state used by the next decision.

F.5 HINDSIGHT TARGET CONSTRUCTION CONTRACT

Target preparation is implemented through state alignment, action selection, and validation. The following box summarizes these checks as a construction contract. It is not an additional free-form Teacher prompt. Its purpose is to make the boundary between privileged guidance and Student-visible input explicit.

Contract for Student-Reachable Hindsight Supervision

```
Input
  Student task and observed prefix
  Student-visible images, tools, and intended Memory setting
  Validated experience and a better verified continuation

Keep
  The first validated recovery action at an aligned state
  Arguments executable with the current task inputs
  Source links needed to inspect the target

Exclude
  Future tool observations from the earlier Student state
  A source answer copied into a tool argument
  Historical file paths or coordinates used without rebinding
  Unexecuted outputs presented as observations
  Duplicate state-action targets

Output
  One admitted state-action example with its provenance
```

For answer targets, the available state must already support the answer. For tool targets, the action must request the evidence that is still missing. A Teacher's eventual success does not change the time at which an observation becomes available to the Student.

F.6 FINALIZATION AND OUTPUT FORMATTING

Finalization uses the observations already acquired by the policy. For a short-answer task, the adapter extracts the requested span or numeric result. For a task that requests calculations or explanation, those fields remain part of the output. A tool budget ending does not justify introducing an unsupported value.

Evidence-grounded finalization, condensed

```
Tool use is finished. Use the question and observations already in
    the conversation to give the best supported response.

Start with the answer declaration required by the task adapter.
    Preserve the requested entities, ordering, units, precision, and
    any explicitly requested calculation or explanation.

Use acquired evidence for names, numbers, and dates. Apply an
    ordinal request to the ordered source rather than to an
    unrelated list. Do not replace a supported value with a guess
    from an earlier draft.

Return the final response without proposing another tool call.
```

G Protocols for the Mechanism Analyses

G.1 Component Comparisons

Table 3 compares Full with four variants on InfographicVQA, VTB, and EB-ALFRED. The variants change different parts of the experience lifecycle. Table 11 states the intervention associated with each name. In particular, removing Internalization means using the Memory-only configuration in this experiment.

Table 11: Definitions of the component comparisons reported in the paper.

Variant	Changed component	Resulting setting
w/o Memory	External retrieval disabled	Adapted policy without inference Memory
w/o Internalization	Parameter adaptation removed	Original policy with process Memory
w/o Visual Attribution	Visual evidence check removed	Candidates passing ordinary audits proceed to abstraction
w/o Router	Selective destination assignment removed	All admitted experience sent to both destinations

The comparison evaluates the resulting system configurations in Section 4.4. Full, Memory-only, and Weight-only share the components specified in Equation 13. Removing the Router sends admitted experience to both destinations and therefore tests the value of selective assignment.

G.2 Visual Evidence and Memory Quality

The visual analysis answers two separate questions. The controlled-input comparison asks whether the proposed region supplies useful content. The downstream comparison asks whether attributed experience improves the resulting Memory under ordinary task inputs. Their denominators differ because the former requires complete intervention inputs and the latter uses the complete evaluation lists.

For each complete intervention task, the evaluator scores the five conditions in Table 9. The plotted bars are averages over the same available task population within each benchmark. The reported figures contain 128 ChartQA tasks and 123 InfographicVQA tasks for the controlled-input analysis. They do not estimate uncertainty from the displayed aggregate values.

The downstream Memory comparison evaluates Naive Memory and Attributed Memory on 200 tasks per dataset. The agent again receives the original task inputs. A gain in this comparison is a property of the complete Memory-assisted configuration, whereas the crop controls describe the local evidence available to a controlled input. One comparison should not be substituted for the other.

G.3 Experience Routing Analysis

This supplementary illustration describes a selection analysis distinct from the component comparison above. The intended analysis compares admitted experiences with random selection from the same candidate pool. Selection quality concerns transfer to independent tasks rather than the Router's own admission score. A complete measurement would bind each selected experience to a fixed training budget and a transfer evaluation that was not used to rank that experience.

Figure 4 provides a synthetic illustration of this supplementary comparison. Each group contains 40 simulated experiences. The boxes show the interquartile range, the central line shows the median, and whiskers extend to the most extreme observations within 1.5 interquartile ranges. The imposed shift illustrates the intended comparison and supplies no additional empirical result. This appendix does not reinterpret those values as measured transfer utility.

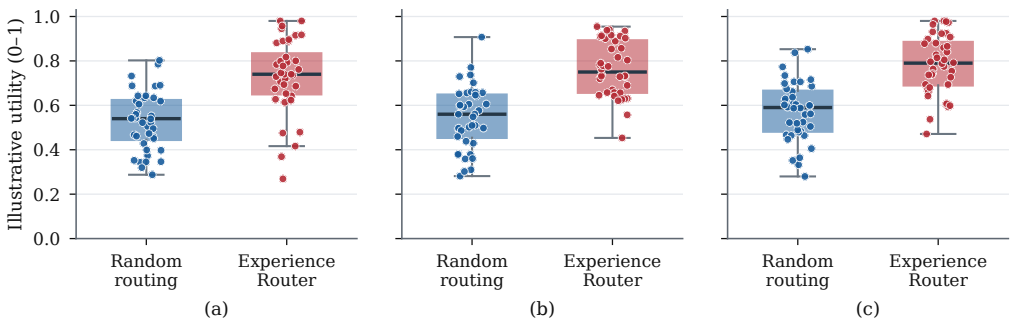

Figure 4: Synthetic routing illustration for (a) InfographicVQA, (b) VTB, and (c) EB-ALFRED. Each group contains 40 simulated experiences. Boxes show the interquartile range, center lines the median, and whiskers the most extreme points within 1.5 interquartile ranges. The imposed distribution shift illustrates a proposed comparison and does not provide experimental evidence of Router effectiveness.

G.4 Does Internalized Experience Survive without Memory?

The claim that experience has been internalized requires evidence that the gain does not depend on reading anything at inference. We therefore evaluate the adapted policy with its Memory withdrawn entirely against the base agent under the same empty context, which leaves the parameters as the only difference, and Figure 5 reports accuracy together with the retrieval and context cost per task.

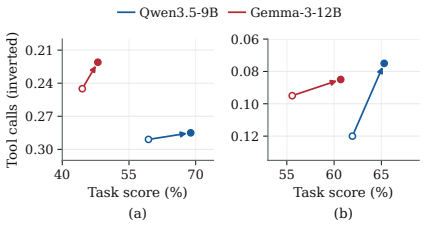

Figure 5: Internalized behavior outperforms the base agent while retrieving nothing, so the gain resides in the policy and not in the retrieved context.

The adapted policy is above the base agent on both tasks and both backbones while retrieving nothing, so the internalized behavior resides in the parameters and not in a context the agent happens to be given. Its cost per task is also lower, which we trace to fewer redundant tool calls once the policy has learned when inspection is warranted. This is the property a memory-only system cannot offer, since every benefit it provides must be re-established through retrieval on each task. The two branches therefore supply persistence of different kinds, one the agent carries with it and one it can consult and revise. Inspecting the records assigned to each destination confirms that the two differ in the kind of content they receive and not only in its quantity, and Appendix K reports this comparison.

Plotting coordinates. Figure 5 groups the two models by dataset. Its axes report task score and mean tool calls per task. Table 12 records the same coordinates.

Table 12: Coordinates used in Figure 5. Scores are percentages. Calls are average tool calls per task. These are the figure's experiment-specific pairs.

Backbone	Dataset	Base score	Weight score	Base calls	Weight calls
Qwen3.5-9B	ChartQA	59.38	68.88	0.291	0.285
Qwen3.5-9B	InfographicVQA	61.94	65.30	0.120	0.075
Gemma-3-12B	ChartQA	44.50	48.00	0.245	0.221
Gemma-3-12B	InfographicVQA	55.57	60.70	0.095	0.085

For task i, let c_i denote the tool-call count in its execution record. The plotted mean is $\bar{c} = N^{-1}\sum_i c_i$. A relative call reduction is computed as $100(\bar{c}_{\text{Base}} - \bar{c}_{\text{Weight}})/\bar{c}_{\text{Base}}$. The vertical axis is inverted so that upward movement corresponds to fewer calls. The horizontal axis retains the task metric of the corresponding dataset, which is relaxed accuracy for ChartQA and ANLS for InfographicVQA.

These pairs document capability and interaction cost under the reported internalization configurations. Their task and configuration identities remain distinct from the historical aggregates in the

main table. Tool calls measure interactions with the exposed tools. They do not directly measure token usage, GPU time, or total computational cost.

H Worked Examples of Experience Use

The examples below illustrate how the method operates on the four task families. They are schematic explanations of the procedures and interfaces, rather than additional measured trajectories. No new accuracy values or task outcomes are inferred from them. Each example follows the same sequence from visible uncertainty to an action, an audit, and a reusable representation.

H.1 Chart Reading and Numerical Comparison

A chart question can fail because the agent reads the wrong series, confuses a category label, or performs arithmetic on an uncertain value. These failures require different corrections. Cropping the relevant region is useful only when the resulting observation retains the context needed to associate the value with the requested category.

Example 1. A chart value is too small to read

The policy first identifies the requested series and category in the original chart. The relevant value remains ambiguous, so it requests a crop that retains the mark and enough surrounding labels to locate it. The returned image is inspected before a number is transcribed. If the question requires a difference or ratio, the operands are associated with their categories before computation.
The audit checks that the crop came from the intended region and that the final computation used supported values. Paired controls can then test whether that region contributed useful content. The reusable procedure is to localize, enlarge, bind labels to values, and verify the operands. The value from this chart is not retained as a general answer. At an earlier state without the crop, the supervised target is the inspection action.

An enlarged mark without its legend may be clear but semantically ambiguous. The recovery rule therefore concerns region selection and context, not simply increasing image size. If the observation is already legible, the procedure should permit a direct answer rather than force a redundant tool call.

H.2 Infographic Text and Layout

An infographic may contain several visually similar numbers or names attached to different panels. OCR can recognize all of them without determining which one answers the question. The policy must preserve the relationship among a heading, a local annotation, and its surrounding visual element.

Example 2. Several panels contain plausible answer strings

The policy identifies the panel associated with the requested concept and inspects its heading and local text together. OCR output is compared with the original layout. A text span from another panel is rejected even when it matches the expected answer type. If the local text remains uncertain, the policy changes the crop boundary or orientation and checks the same semantic relation again.
The experience record retains the applicability condition and the procedure for linking text to its panel. It does not store the answer string as a transferable fact. During consolidation, repeated support for this behavior can justify an internalized decision to inspect the relevant local context. Student supervision still uses only the images and observations available at the selected state.

The answer formatter returns the span requested by the task. Completeness and factual support are separate checks. Supplying a well-formed name does not validate that it came from the correct panel, and a correct local reading does not justify unrelated additional text.

H.3 Visual Measurement with Code

Tool-enabled visual reasoning often depends on an intermediate representation. A grid, mask, set of components, or coordinate array can make a computation possible, but the representation itself must be checked against the image. A program that runs successfully can still solve the wrong geometric problem.

Example 3. A count depends on image segmentation

The policy states what should be counted and inspects the relevant region and image dimensions. It executes the measurement code and retains both the returned quantity and any visual artifact needed to inspect the segmentation. Before answering, it checks whether the detected components correspond to the requested objects. If one object is split into several components, the segmentation or counting rule is revised.
The execution audit records whether the program completed. The visual-support audit checks whether its operands and intermediate representation correspond to the image. The retained procedure specifies this validation sequence and its recovery rule. It does not promote the first numeric output merely because no exception was raised. Parameter supervision teaches the appropriate next measurement or verification action at the state where the uncertainty occurs.

This example distinguishes computational correctness from perceptual grounding. Arithmetic can be exact while the selected region or representation is wrong. The experience should capture the decision that corrects this failure mode, including the condition under which another observation is necessary.

H.4 Embodied Preconditions and Recovery

In EB-ALFRED, a correct action name can fail because a prerequisite is absent. The agent may not be holding an object, the target may not be visible, or a receptacle may not be ready for placement. Repeating the same action without changing the failed condition is unlikely to provide new evidence.

Example 4. Placement fails after an earlier interaction

The policy uses the current image and environment feedback to determine whether it still holds the required object. It identifies the intended receptacle and satisfies the relevant visibility or opening condition before placing the object. After the action, it checks the returned feedback and updates the tracked state. Dropping an object is not treated as completing a placement goal.
An audited record links the failed interaction to the missing precondition and the successful recovery sequence. The reusable procedure describes how to check possession and establish a valid placement target. Historical object identifiers and scene locations are rebound to the current state. A Student-visible failure state can then support supervision of the next recovery action without supplying a future observation or a complete successful trajectory.

The same logic applies to tasks requiring heating, cooling, cleaning, or multiple objects. The experience encodes when an operation is required and which feedback confirms progress. It does not treat an action-menu entry or an intended action as evidence that the corresponding state change has already occurred.

I Implementation and Supplementary Artifacts

I.1 Component Responsibilities

The implementation separates interaction, experience storage, attribution, supervision preparation, training, and evaluation. Table 13 maps the paper components to the corresponding source modules. Paths are relative to the supplied project and do not depend on a particular server directory.

Table 13: Implementation map for the VisEvolver experience lifecycle.

Component	Source and responsibility
Typed records	`src/visevolver/schemas.py` defines tasks, episodes, evidence, procedures, and supervision records.
Interaction	`src/visevolver/agent/` prepares model inputs and preserves the action-observation sequence.
Tool execution	`src/visevolver/tools/` executes image and code operations and records outputs and errors.
Visual attribution	`src/visevolver/attribution/` constructs visual controls and links evidence to source decisions.
Experience abstraction	`src/visevolver/evolution.py` and `src/visevolver/pipeline.py` build reusable records from supported experience.
External Memory	`src/visevolver/memory/` stores records, retrieves candidates, and renders compact context.
Parameter admission	`src/visevolver/distillation/gate.py` evaluates experience eligibility for training.
Target preparation	`src/visevolver/distillation/prepare.py` and `src/visevolver/distillation/samples.py` align states and validate targets.
LoRA and preservation	`src/visevolver/distillation/train_lora.py` implements action supervision and preservation.
Task adapters	`src/visevolver/paper_suite/` and benchmark-specific modules bind the selected task interfaces and assets.

The implementation contains both experience-admission logic and runtime retrieval controls. The former determines which supported behavior may supply parameter supervision. The latter controls the context supplied to a particular inference task. They contribute to the complete experience pipeline but should not be interpreted as one additional learned policy that makes every decision.

I.2 Configuration and Asset Identity

A reproducible execution binds a task population to a backbone, adapter, Memory snapshot, prompt configuration, tool interface, and evaluator. The model name alone does not identify the complete system. An adapter must be paired with its intended base model and chat template. A Memory snapshot must identify the procedures available at the time of evaluation.

Task records retain the sample identifier, source image or environment initialization, split role, and rollout seed. Tool records retain their arguments, return status, and produced artifacts. Supervision records additionally link the Student state, target action, and supporting experience. These identities connect a compact process record to its source without placing the entire source trajectory into the deployed prompt.

Run and asset record, abbreviated organization

```
task_population
  benchmark, task_ids, split, source identities
model
  backbone, revision, chat template, decoding configuration
deployment
  adapter identity, Memory snapshot, enabled tools, task budget
execution
  task and seed, actions, observations, errors, final output
supervision
  visible state, admitted target, source experience identities
evaluation
  metric, scored population, per-task outputs, aggregation rule
```

A configuration record and its score serve different purposes. The former identifies what was executed, and the latter summarizes its outcomes. Preserving both makes it possible to compare the intended Memory and parameter settings without reconstructing them from rounded table values.

I.3 Supplementary Package Organization

The supplementary package contains the core implementation, selected benchmark-specific runtime sources, configuration templates, and small dataset examples. The examples include two tasks from each of ChartQA, InfographicVQA, VTB, and EB-ALFRED. They illustrate data formats and input bindings. They are not a substitute for the evaluation populations in Table 4.

Supplementary package layout

```
src/visevolver/       Core interaction and experience lifecycle
configs/              Configuration templates and prompt files
scripts/              Inspection and demonstration entry points
examples/
  chartqa/            Two task and image examples
  infographicvqa/     Two task and image examples
  vtb/                Two task and image examples
  eb_alfred/          Two task and trajectory-annotation examples
reference/            Selected benchmark-specific source snapshots
SOURCE_MANIFEST.json  Source provenance and path substitutions
FILES.sha256          Checksums for the distributed files
```

The compact package does not bundle model weights, trained adapters, complete Memory databases, full datasets, or external simulator assets. Real training and evaluation require the corresponding assets and benchmark dependencies. The EB-ALFRED examples provide task and trajectory annotations, while actual visual observations are produced by the environment. Source-specific dataset and dependency licenses remain applicable.

I.4 Inspecting the Implementation

The package can be inspected without running a new benchmark experiment. The CPU demonstration exercises the typed records and experience lifecycle using synthetic fixtures. It does not train a LoRA or produce any result reported in the paper. The commands below describe inspection of the supplied package.

Package inspection and demonstration

```
python -m pip install -r requirements-demo.txt
python scripts/inspect_examples.py
python scripts/demo_core.py

PYTHONPATH=src python -m visevolver.cli attribute-run --help
PYTHONPATH=src python -m visevolver.cli evolve-memory --help
PYTHONPATH=src python -m visevolver.cli distill-prepare --help
PYTHONPATH=src python -m visevolver.cli distill-train --help
PYTHONPATH=src python -m visevolver.cli rollout --help
```

For model-backed execution, the separate stages collect rollouts, construct attribution evidence, consolidate Memory, prepare supervision, train an adapter, and evaluate frozen assets. Each stage consumes explicit records from the previous stage. This organization permits inspection of the decision that admitted an experience without treating a later task score as a replacement for the original evidence.

I.5 Tool and Artifact Handling

Image tools retain their output dimensions and applied transformations. Identical image artifacts can be stored by content identity, while distinct observations retain separate source links. Code execution receives explicitly bound task inputs and returns its status and generated outputs. A source path is an artifact binding, not visual evidence by itself.

Runtime outputs and audit metadata remain separate from the instructions rendered to the policy. Raw episodes preserve detailed observations, while deployment context uses the compact procedure representation. This division supports both a readable policy input and an inspectable experience history, matching the Memory and parameter roles described in the main method.

J Limitations and Future Work

VisEvolver audits visual support at the level of image regions, which fits tasks whose evidence is localized and leaves open how to attribute experience whose support is distributed across an entire scene or across several frames. The routing criteria read five signals whose thresholds express how much support an experience must accumulate before it may change parameters, and replacing these fixed thresholds with a criterion adapted to the observed experience distribution is a natural extension. Internalization currently updates the language-side policy only, and extending the same audited supervision to the perceptual pathway would allow the agent to improve what it sees in addition to how it uses what it sees.

K What Kind of Experience Reaches Each Destination

To understand what the Router separates in practice, we inspect the records assigned to each destination over the training split of both backbones. Records that reach the parameters describe conditions for acting, such as enlarging a region before reading a value when the text is below a legible size, or rechecking a conflicting label before committing to an answer. Records that stay in Memory name task-specific constraints, such as the argument format a tool accepts or the order in which two operations must be applied. Records that reach both destinations pair a general habit with a constraint that would be unsafe to absorb, so the policy acquires the habit while the constraint remains editable. The separation follows the signals of Eq. 8, where cross-task support and low variance mark the first group and high instance specificity marks the second. The destinations therefore differ in the kind of content they receive, and a fixed form of persistence has no way to express that difference.