YouRA: A Persistent-State Architecture for Evidence-Traceable Autonomous Research Agents

Yoonkyu Woo[1,2*] **Woojin Lee**[1*] **Jin-xia Huang**[1,2†]

[1]Electronics and Telecommunications Research Institute, Republic of Korea
[2]Department of Artificial Intelligence, University of Science and Technology, Republic of Korea
{yoon303, writerwoody, hgh}@etri.re.kr

Abstract

End-to-end research agents can now produce complete scientific papers, yet manuscript claims often diverge from executed experiments. This gap is structural: research state, failure histories, and claim–evidence alignment are not maintained as persistent, verifiable state across long-horizon pipelines. We present **YouRA (Your Research Agent)**, an architecture for stateful, evidence-traceable autonomous research. YouRA preserves research state, execution evidence, and failure history across the research trajectory by integrating three components: a *Verification State Architecture* (VSA) that tracks hypotheses, gates, and evidence pointers; an *Independent Controller* that turns state and reflection records into lifecycle, recovery, and debate/review control while separating control from execution; and *Stateful Reflection* that logs failures as structured lessons and routes recovery through bounded repair, redesign, or reset. On MLR-Bench's predefined ten-task end-to-end subset, YouRA improves over both MLR-Agent and AI Scientist V2 on scalar Overall across all three matched backbones. An automated diagnostic using MLR-Bench's hallucination taxonomy reports intersection/union counts for four fact-based failure types, and data-provenance diagnostic shows more real-data-based outputs. Ablating each of the four components (the VSA, the Independent Controller, MCP tool access, and reflection-guided recovery) supports their separable contributions. Removing either core-state component drops YouRA below the full system. Code: https://github.com/PrayPrey/Your-Research-Agent.

1 Introduction

Autonomous research agents that formulate hypotheses, run experiments, and write papers have been pursued for decades, from closed-loop experimentation (King et al., 2009) through proposals for AI-amplified discovery (Gil et al., 2014) to explicit Nobel-level challenges (Kitano, 2021). Large language models have made this vision practical: AI Scientist (Lu et al., 2024) and its tree-search successor V2 (Yamada et al., 2025) automate the full pipeline from ideation to paper writing, AI-Researcher (Tang et al., 2025) demonstrates role-separated multi-agent collaboration, Agent Laboratory (Schmidgall et al., 2025) integrates human feedback, and VIRSCI (Su et al., 2025) and AI Co-scientist (Gottweis et al., 2025) refine ideas through multi-agent debate. Context-management strategies differ across these systems, but none externalizes the research trajectory as explicit, persistent, verifiable state spanning the full pipeline. That gap limits reliability over long horizons.

This gap appears in three concrete limitations on the reliability of automatically generated research. (1) **Context and evidence fragmentation.** When research state is implicitly embedded in conversation histories, it is vulnerable to truncation and hallucination, which breaks long-horizon coherence across hypotheses, experiments, conclusions, and manuscript claims (Yan et al., 2026). Current agents also provide only limited mechanisms for ensuring that manuscript claims remain aligned with executed code, result files, and logs. As a result, a paper can contain superficially persuasive claims generated from hallucinated results or failed runs, and such claims may still appear strong under output-only judging that does not inspect the underlying artifacts. (2) **Lack of structured failure memory.** Agents lack mechanisms to record, diagnose, and learn from experimental failures across stages. MLR-Bench's hallucination analysis reports that evaluated coding agents produced fabricated or unverified experimental results in roughly 80% of cases, with case studies tracing such failures to execution errors masked by simulated out-

[*]Co-first Authors
[†]Corresponding Author

Proceedings of the 15th International Joint Conference on Natural Language Processing and the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 355–375
November 6-10, 2026 ©2026 Association for Computational Linguistics

puts (Chen et al., 2026). (3) **Unreliable workflow control.** In long-horizon research agents, the same task-execution dialogue often both performs the work and decides how the workflow should continue. It carries code-writing and paper-writing context together with control decisions such as when to stop, retry, redesign a hypothesis, or move to manuscript review. As this dialogue grows, these control decisions become vulnerable to truncation, drift, and inconsistent recall, making it difficult to resume or recover a run from the actual research state.

All three limitations trace back to the same absence: the research trajectory is not maintained as explicit, persistent, verifiable structure across the pipeline. Two pieces are missing. One is a shared state substrate that records hypotheses, gates, failures, and evidence pointers. The other is an external control surface that uses this state to resume, redirect, and review the research lifecycle.

To address these problems, we present **YouRA (Your Research Agent)**, an architecture for stateful, evidence-traceable autonomous research that preserves research state, execution evidence, and trajectory-level failure history as explicit, persistent structure (Figure 1). Its three components are: (i) a **Verification State Architecture (VSA)** that centrally manages hypotheses, gates, and evidence pointers; (ii) an **Independent Controller**, which consumes VSA and reflection records to drive lifecycle transitions, recovery escalation, and multi-agent debate/review roles while separating high-level control from task execution; and (iii) **Stateful Reflection**, which logs failures as structured lessons and routes recovery through bounded repair, redesign, or reset. Given only a topic-level description, YouRA autonomously expands it into a research question, hypothesis, experiment plan, implementation, validation evidence, and final manuscript. In practical research use, the same architecture supports dynamic researcher steering through stage-specific CLI commands for redirecting, restarting, or redesigning a run; for evaluation as an autonomous research agent, however, all benchmark runs in this paper are held to fully autonomous mode.

We evaluate on MLR-Bench (Chen et al., 2026), a benchmark designed for open-ended ML research agents that provides 201 workshop-derived research tasks, stepwise and end-to-end evaluation protocols, and structured MLR-Judge rubrics. It directly matches YouRA's problem setting because it evaluates the full path from ideation through experimentation to paper writing, and explicitly surfaces fabricated or unverified experimental results as a central reliability failure. Following the benchmark's execution-heavy paper-level setting, our end-to-end and pairwise comparisons use its predefined ten-task subset. On this subset, YouRA obtains the highest 4-judge end-to-end Overall among all systems on every matched backbone, outperforming both MLR-Agent and AI Scientist V2 (Yamada et al., 2025). Order-controlled pairwise preferences also favor YouRA over both baselines on every reported backbone. Beyond the rankings, an automated diagnostic based on MLR-Bench's hallucination taxonomy decomposes concrete failure modes across matched backbones, and the data-provenance diagnostic across matched backbones shows that YouRA outputs more often rely on real data. Ablations of each of the four components support their separable contributions. Ablations of all four components significantly reduce Overall, while pairwise comparisons of their drop magnitudes do not support a reliable ranking among components.

YouRA's contribution is to make research state, execution evidence, and the research trajectory explicit, durable objects in autonomous research rather than byproducts of transient dialogue context: hypotheses, failures, evidence pointers, and recovery decisions persist in shared state and Reflective Memory, where later agents can read them. VSA-based state externalization supplies the record, the Independent Controller resumes, redirects, and reviews the lifecycle from that record while keeping control separate from execution, and Stateful Reflection drives failure-based recovery. YouRA also carries an evaluation perspective: an autonomous research system should be judged on whether its final claims remain traceable to executed evidence and the research trajectory that produced them, and a persuasive manuscript alone should not settle that judgment.

2 Related Work

End-to-End Autonomous Research. Recent end-to-end systems fall along three architectural axes. *Single-pipeline* designs chain ideation through paper writing into a fixed flow (Lu et al., 2024); V2 extends exploration depth via agentic tree search (Yamada et al., 2025). *Role-separated multi-agent* designs specialize agents per stage

(Tang et al., 2025), but do not make a shared persistent state the coordinating substrate for long-horizon coherence. *Human-feedback-integrated* designs accept researcher feedback at stage transitions (Schmidgall et al., 2025), but restrict intervention to predefined checkpoints. Across these approaches, the research trajectory is still not carried between stages as durable, verifiable state, and failures never become structured constraints on later work. This is the architectural gap behind the three limitations identified in the Introduction.

Collaborative Systems and Persistent State. A parallel line of work explores human–AI collaboration for research quality. *VIRSCI* (Su et al., 2025) refines hypotheses through team-based multi-agent discussion, and *AI Co-scientist* (Gottweis et al., 2025) combines Elo-ranked tournament evolution with researcher guidance, yet both stop at the pre-experimentation ideation stage. Enterprise Deep Research (Prabhakar et al., 2025) introduces steerable multi-agent deep research with human-in-the-loop guidance for analytical report generation, but targets information synthesis and does not run hypothesis-testing pipelines with code execution. YouRA targets a different setting: hypothesis-testing pipelines where state must be preserved across ideation, implementation, failure recovery, and writing. The most directly relevant mechanism for learning from failures is Reflexion (Shinn et al., 2023), which applies per-task verbal reinforcement within an episodic memory buffer scoped to a single task. YouRA extends this principle to a structurally different regime. Failure lessons persist across heterogeneous research stages, from hypothesis redesign to full research-question reset, and are coordinated through the VSA instead of being held in a single-task buffer. For hypothesis generation, YouRA builds on the multi-agent scientific-debate paradigm from VIRSCI (Su et al., 2025) and AI Co-scientist (Gottweis et al., 2025), but couples it to VSA state and Reflective Memory so debate outputs remain tied to downstream verification.

Evaluation and Reliability. Research-agent reliability benchmarks now span multiple scopes. MLAgentBench (Huang et al., 2024) measures ML experimentation success. DeepResearch Bench (Du et al., 2026) targets report quality; FML-bench (Zou et al., 2026) evaluates exploration breadth; and MLR-Bench (Chen et al., 2026) assesses the full ML-research pipeline from ideation to paper writing. MLR-Bench demonstrates that agents can mask execution failures with simulated or placeholder results, exposing a gap between fluent paper generation and trustworthy experimentation. Failure analyses trace related reliability problems to weak factual grounding and insufficient intermediate verification (Zhang et al., 2025). YouRA targets this gap: it preserves failures as structured records that constrain subsequent stages, and embeds evidence-checking mechanisms before and after paper generation (§3).

Limitations of Output-Only Benchmarks. Output-only evaluation of final papers has known limits: reference-free LLM judges show high agreement with human experts only on questions the judges themselves can answer (Krumdick et al., 2026), paper–code alignment benchmarks reveal that even top models detect fewer than half of real discrepancies between papers and their implementations (Baumgärtner and Gurevych, 2026), and coding agents lose implementation fidelity when task specifications emerge incrementally over long horizons (Yan et al., 2026). These limitations connect directly to the first problem raised in the Introduction, context and evidence fragmentation: protocols that evaluate only the final paper can structurally miss gaps between manuscript claims and executed experiments (Liu and Zhu, 2026). We therefore use MLR-Bench's hallucination taxonomy to inspect concrete fact-based failure types in final papers, and add a separate data-provenance diagnostic that summarizes whether manuscripts rely on real, synthetic, or fabricated artifacts. These diagnostics complement the primary benchmark rankings by exposing claim–evidence audit signals that scalar output scores can miss (§5).

3 Methodology

Having identified the absence of persistent research state as the common root of these limitations (§1–2), we now describe how YouRA addresses each one through a dedicated architectural component. These structural mechanisms are intrinsic to YouRA's architecture and cannot be grafted onto existing research agents as standalone modules; the individual effect of each component is analyzed in detail in the ablation studies (§4.3). Here, *state externalization* means storing research state (hypotheses, gates, failures, and evidence pointers) in structured files instead of conversation history, so each lifecycle stage starts from a stable record.

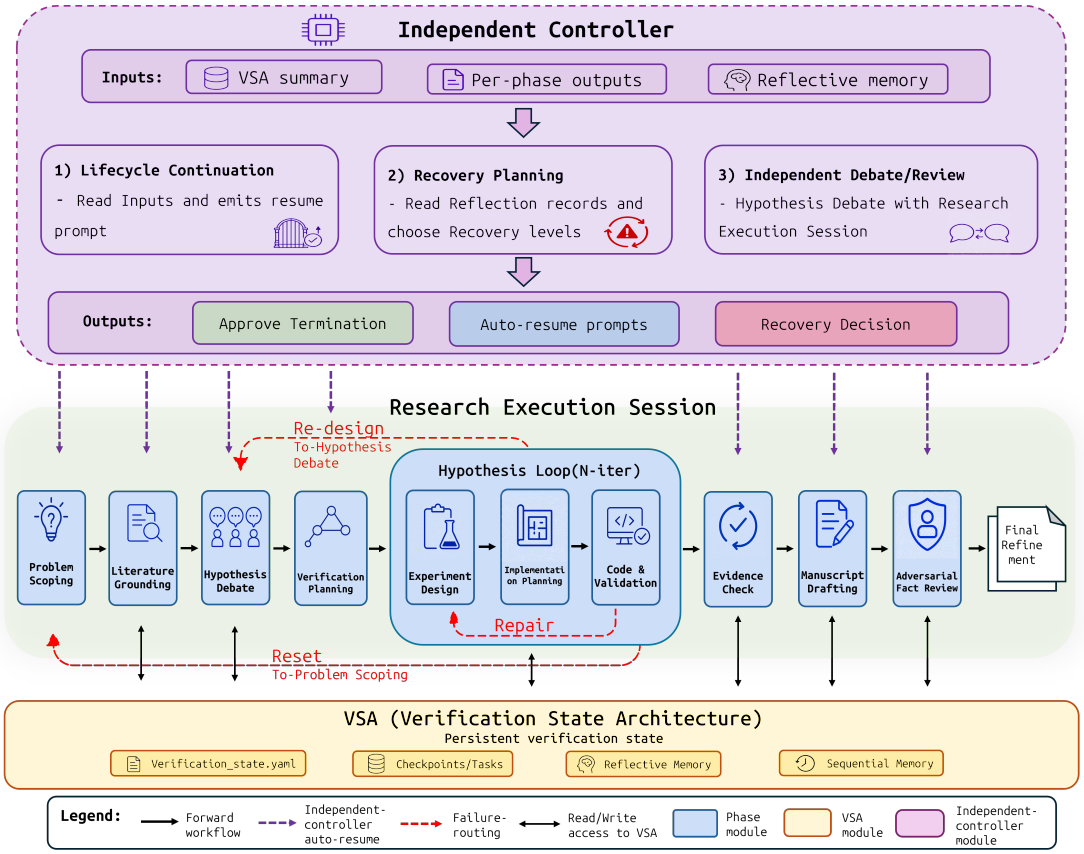


Figure 1: YouRA research lifecycle. Stages proceed left to right under the *Verification State Architecture (VSA)* (§3.2). An independent controller LLM manages lifecycle transitions and persona selection. Sub-hypotheses traverse the hypothesis loop, and dashed arrows denote reflection-guided three-level routing (repair, redesign, and reset) for failed gates (§3.4). The figure uses the reader-facing stage names introduced in §3.1.

YouRA implements this through three components that together address the three limitations from §1: the *Verification State Architecture* (§3.2) addresses context and evidence fragmentation by preserving hypotheses, gates, and evidence pointers; the *Independent Controller* addresses unreliable workflow control by consuming VSA and reflection records to drive lifecycle transitions, recovery routing, and multi-agent hypothesis debate (§3.3); and *Stateful Reflection* (§3.4) addresses the lack of structured failure memory through three-level failure routing. In implementation, procedural context such as reusable stage templates, reference documents, and modular skills is linked by explicit file pointers, so each stage loads the relevant context on demand. Figure 1 illustrates the full lifecycle, and §3.1 summarizes the stage sequence, including the evidence-checking and refinement passes that connect experimental execution to manuscript writing.

3.1 Collaborative Research Lifecycle Overview

The workflow comprises ten named stages under VSA-controlled transitions (Figure 1). Problem Scoping and Literature Grounding turn a broad interest into grounded research gaps; Hypothesis Debate uses a six-persona debate protocol to converge on a hypothesis; and Verification Planning converts that hypothesis into a dependency-aware directed acyclic graph (DAG) plan and gated sub-hypotheses. The subsequent *Hypothesis Loop* iterates through Experiment Design, Implementation Planning, and Coding & Validation, turning each sub-hypothesis into an experiment specification, implementation plan, and validated code. Evidence Check records which predictions and assumptions are consistent with executed results before Manuscript Drafting uses a section-grouped drafting protocol (Story Group Architecture; Appendix A) to produce the paper. Adversarial Fact Review checks the generated manuscript against the VSA and accumulated stage documents, and a

final Manuscript Refinement pass integrates the review output with prior artifacts, results, logs, and code to improve logical consistency, evidential alignment, and narrative clarity. Stage-level mechanics and recovery decision logic are detailed in Appendix A.

3.2 Verification State Architecture

YouRA's first component targets context and evidence fragmentation. In conventional agent loops, state is implicitly embedded in conversation history, where it is vulnerable to truncation and hallucination. YouRA instead maintains research state through a YAML-based *Verification State Architecture (VSA)*. A central `verification_state.yaml` tracks macro pipeline state (hypotheses, gate results, routing decisions, evidence pointers, and a failed-hypotheses registry), and per-stage checkpoint files complement it. Later stages read gate outcomes, evidence pointers, and failure records from the VSA, so executed experiments become explicit constraints when hypotheses are revised, validated, or written into manuscript claims. The VSA is a *single persistent state record* under three invariants: *gate-driven progression* (a hypothesis advances only when its `MUST_WORK` gates are satisfied), *auditable history* (every mutation is timestamped and attributed), and *steering as mutation* (researcher directives become VSA field updates).

3.3 Independent Controller

YouRA's second component turns persistent state into control. The VSA stores the research trajectory, and Stateful Reflection records failure constraints and routing history; the independent controller consumes these records to choose the next action. Concretely, the controller supports three roles: **lifecycle continuation**, selecting the next stage from VSA gate status; **recovery escalation**, consuming Stateful Reflection records to choose repair, redesign, or reset boundaries; and **independent debate/review**, used by Hypothesis Debate and Adversarial Fact Review. By keeping control in a separate context from task execution, YouRA avoids making the execution conversation itself the only source of control state. The controller therefore reads compact VSA/artifact summaries rather than the full trace and emits continuation context at stop or human-in-the-loop boundaries.

Prior systems such as VIRSCI and AI Co-scientist provide functional precedents for multi-agent scientific debate in hypothesis generation (Su et al., 2025; Gottweis et al., 2025). YouRA does not adopt their pipelines; it implements the analogous debate/review function inside its own stateful architecture, through a Hypothesis Debate protocol tied to the VSA and Reflective Memory. Six research personas debate candidate hypotheses while the controller LLM and task-execution LLM alternate persona contributions; Reflective Memory turns refuted approaches into constraints and validated or partially validated elements into reusable evidence. The debate terminates once specificity, mechanism clarity, testable predictions, novelty, feasibility, and objection resolution are jointly satisfied; the converged hypothesis is formalized and decomposed into 3–7 gated sub-hypotheses in the VSA.

3.4 Stateful Reflection

YouRA's third component, **Stateful Reflection**, turns failure signals into structured constraints that later stages can use. It operates over Reflective Memory (§3.2) during the hypothesis loop: routing-relevant outcomes are recorded with structured reasons, supporting or refuting evidence, and pivot-or-reuse rationales, so refuted approaches become negative constraints while validated and partially validated components become reusable evidence for successor stages. Recovery proceeds in three levels. Bounded repair first handles local Coding & Validation failures, and after each execution an independent mock-data and hard-coded-result detector checks whether reported outputs are derived from actual data processing; detected fabrications trigger repair before evidence is passed downstream. If a `MUST_WORK` gate remains unsatisfied after repair, Stateful Reflection escalates to Hypothesis Debate redesign when mechanism revision remains recoverable, or to Problem Scoping reset when the current hypothesis contradicts executed evidence. Each sub-hypothesis terminates when its gates resolve through validation, refutation, or escalation, after which the final state is recorded and the pipeline proceeds to Evidence Check. Routing telemetry is reported in Appendix Table 9 and Appendix C.

4 Experiments

4.1 Setup

Benchmark. We evaluate on MLR-Bench (Chen et al., 2026), a benchmark for open-ended ML research agents that provides 201 workshop-derived research tasks, MLR-Judge rubrics, and the MLR-

System	Backbone LLM	Clarity	Novelty	Soundness	Significance	Overall
MLR-Agent	Sonnet 4.5	7.42 ± 0.24	5.12 ± 0.74	2.48 ± 0.61	3.52 ± 0.68	3.10 ± 0.64
	Opus 4.5	7.30 ± 0.35	5.25 ± 0.31	3.05 ± 0.96	3.80 ± 0.71	3.62 ± 0.80
	Sonnet 4.6	$\mathbf{7.78 \pm 0.22}$	$\mathbf{5.95 \pm 0.44}$	4.05 ± 1.19	4.72 ± 1.14	4.42 ± 1.11
AI Scientist V2	Sonnet 4.5	6.72 ± 0.43	4.85 ± 0.60	3.45 ± 1.18	3.92 ± 0.85	3.62 ± 0.84
	Opus 4.5	6.38 ± 1.25	5.22 ± 0.86	4.18 ± 1.31	$\mathbf{4.47 \pm 1.03}$	4.28 ± 1.16
	Sonnet 4.6	6.45 ± 0.65	5.75 ± 0.47	3.32 ± 1.01	4.22 ± 0.95	3.88 ± 0.88
YouRA	Sonnet 4.5	$\mathbf{7.50 \pm 0.49}$	$\mathbf{5.23 \pm 0.92}$	$\mathbf{3.98 \pm 1.74}$	$\mathbf{4.30 \pm 1.28}$	$\mathbf{4.20 \pm 1.45}$
	Opus 4.5	$\mathbf{7.73 \pm 0.48}$	$\mathbf{5.33 \pm 1.03}$	$\mathbf{4.58 \pm 1.86}$	4.33 ± 1.42	$\mathbf{4.45 \pm 1.51}$
	Sonnet 4.6	7.72 ± 0.32	5.70 ± 0.57	$\mathbf{4.92 \pm 1.09}$	$\mathbf{5.05 \pm 1.00}$	$\mathbf{5.05 \pm 0.92}$

Table 1: MLR-Bench end-to-end scores (10 tasks; 1–10, higher is better). Each cell reports the mean ± task SD of scores averaged over four judges; bold marks the best score per matched-backbone block.

Comparison	Backbone	Win	Tie	Lose
YouRA vs MLR-Agent	Sonnet 4.5	25	14	1
	Opus 4.5	26	5	9
	Sonnet 4.6	21	14	5
YouRA vs AI Scientist V2	Sonnet 4.5	18	14	8
	Opus 4.5	17	12	11
	Sonnet 4.6	20	12	8

Table 2: Order-controlled pairwise preferences from the YouRA perspective. Each row summarizes 40 order-collapsed judge-task verdicts (10 tasks × 4 judges); each verdict is scored in both orders, with reversals counted as ties.

Agent baseline pipeline for idea generation, proposal formulation, experimentation, and paper writing. The 201 tasks are sourced from recent ICLR/ICML/NeurIPS workshops and are categorized by MLR-Bench into nine core ML topics: LLMs and Vision-Language Models, AI for Science, ML Theory, Trustworthy AI, Computer Vision, ML Systems, Multimodality, Reinforcement Learning, and other emerging topics. End-to-end and pairwise paper-level evaluation use the predefined ten-task subset specified by MLR-Bench (Table 1 and Table 2). As supplementary upstream context on the full task pool, Appendix Table 13 compares YouRA and MLR-Agent on Idea/Proposal-stage quality across all 201 tasks.

Systems. We compare three systems under Claude Sonnet 4.5 (Anthropic, 2025b), Opus 4.5 (Anthropic, 2025a), and Sonnet 4.6 (Anthropic, 2026b) backbones: (1) **MLR-Agent** (Chen et al., 2026), the reference scaffold released with MLR-Bench, (2) **AI Scientist V2** (Yamada et al., 2025), the benchmark's primary external end-to-end comparison system and (3) **YouRA** (ours).

Following the MLR-Bench convention (Chen et al., 2026), we run each system in its released configuration: MLR-Agent's default scaffold, which invokes AI code editors such as Codex[1] and Claude Code[2] but does not integrate a separate MCP stack; AI Scientist V2 with Semantic Scholar API access and YouRA with the Model Context Protocol (MCP) tool-access stack (Model Context Protocol Contributors, 2025) summarized in Appendix Table 6. This is a released-configuration comparison with matched research-execution backbones, not a fully tool-normalized comparison; architecture-level interpretation therefore relies on the within-YouRA ablations and the provenance/hallucination diagnostics in addition to the cross-system scores. For YouRA, the Claude model is the task-execution backbone and the independent controller is fixed as GPT-5.2 (OpenAI, 2025) (removed only in the w/o Independent Controller ablation. §4.3), so Table 1 varies execution rather than controller backbone.

Evaluation. We adopt MLR-Bench's End-to-End rubric (Chen et al., 2026) (Clarity, Novelty, Soundness, Significance, Overall; 1–10): **Clarity** scores readability and structure, **Novelty** scores originality and advance over prior work, **Soundness** scores method appropriateness, data support, code consistency, and whether reported results are real, not fabricated, **Significance** scores whether reliable results support field-level impact, and **Overall** gives a holistic publishability-anchored judgment. We retain these evaluation criteria and adapt MLR-Judge's input handling to account for each

[1] https://github.com/openai/codex
[2] https://github.com/anthropics/claude-code

judge LLM's context length limit. Each judge receives the task description, paper, code, result files, and experimental logs. Across all evaluated systems, when a request exceeded this limit, we progressively truncated the combined code, result files, and logs from the end and retried the request, while keeping the task description and paper in full. The MLR-Bench study reports alignment between MLR-Judge scores and human judgments, so we use this rubric as the primary standardized metric while acknowledging the general limitations of automated judging. We therefore complement scalar scores with order-controlled pairwise preferences, hallucination diagnostics, data-provenance diagnostics, and component ablations. End-to-end scoring on the ten-task subset averages four frontier-class judges from distinct model families (Gemini 3.1 Pro (Google DeepMind, 2026), GPT 5.4 (OpenAI, 2026), Grok 4.3 (xAI, 2026), and Opus 4.6 (Anthropic, 2026a)), selected so that judge capability meets or exceeds that of the generators under evaluation while limiting judge–generator overlap. Pairwise win rates use the same four judges with order-bias control: for each paper pair, we evaluate both presentation orders and count the comparison as a tie if the preferred paper changes when the order is reversed. All evaluations run fully autonomously.

4.2 Main Results

Upstream stages. We first check that YouRA does not win simply by starting from better ideas. Scoring both systems' Idea and Proposal outputs on the full 201-task pool with a single Gemini 3.0 Pro judge shared across systems (Appendix D.2), we find YouRA lower or statistically indistinguishable on Overall on both backbones measured: MLR-Agent scores higher with Sonnet 4.5, significantly so at the Idea stage, and the two do not differ with Opus 4.5. Where we can compare, the end-to-end advantage reported below therefore cannot be traced to a head start at ideation or proposal writing. It emerges over the later span of the workflow, consistent with the gains arising in execution, verification, and writing rather than in idea quality.

End-to-end scores. Table 1 gives the scalar results of the end-to-end evaluation. YouRA is the only system that ranks first in *Overall* on all three matched backbones, and the gains are concentrated in dimensions tied to research validity. Against MLR-Agent, YouRA improves Overall on every backbone and shows the largest separation in *Soundness* and *Significance*. It also leads AI Scientist V2, the strongest prior end-to-end baseline included by MLR-Bench, on Overall across backbones and achieves higher *Soundness* in all three settings. Both Overall gaps are significant under a paired sign-flip permutation test over the 30 matched (backbone, task) cells (Appendix D.1).

Pairwise preferences. Table 2 reports pairwise preferences, with each pair judged in both presentation orders to control position bias (Zheng et al., 2023). Across the three backbones (120 comparisons per baseline), YouRA wins 72, ties 33, and loses 15 against MLR-Agent, and wins 55, ties 38, and loses 27 against AI Scientist V2. It comes out ahead on every backbone, so the advantage holds even when judges compare two papers from the same task side by side. An automated hallucination diagnostic with human validation complements these score-based results (§5.1).

4.3 Ablation Studies

Table 3 reports four component ablations of YouRA across all three backbones, each dropping one component from the full system. *w/o MCP* removes the MCP tool layer along with tool-grounded execution and structured access to memory and external artifacts. *w/o Reflection* disables reflection-guided redesign and reset, so failures and partial successes no longer turn into reusable constraints and evidence. *w/o VSA* swaps the shared persistent record of hypotheses, evidence, gate outcomes, and failure constraints for a short in-context state summary at each stage. *w/o Independent Controller* moves stage progression, recovery, and debate convergence into the research execution session, where they become simple artifact checks. The drops are largest on *Soundness* and *Significance* and smaller on *Clarity* and *Novelty*, which means the components matter mainly for research validity, not writing clarity or assessed novelty. Removing either core-state component, the VSA or the Independent Controller, also pushes YouRA's Overall below both baselines on Sonnet 4.5 and Opus 4.5 (Tables 1 and 3).

Across the 30 matched backbone–task cells, every ablation significantly lowers *Overall* under Holm-corrected paired sign-flip permutation tests. (Appendix G) Comparing the drop magnitudes pairwise does not turn up statistically reliable differences among the components. These results sup-

Table 3: Component ablations on MLR-Bench (10-task subset, four-LLM-judge panel, mean ± sample standard deviation). The VSA, IC (Independent Controller), MCP, and Refl. columns indicate whether each component is enabled in a configuration. All four ablations are run on Sonnet 4.5, Sonnet 4.6, and Opus 4.5. Rows report raw mean scores. Drops from the full configuration are discussed in the text.

Configuration	Backbone	VSA	IC	MCP	Refl.	Clarity	Novelty	Soundness	Significance	Overall
YouRA (full)	Sonnet 4.5	✓	✓	✓	✓	7.50 ± 0.49	5.23 ± 0.92	3.98 ± 1.74	4.30 ± 1.28	4.20 ± 1.45
	Sonnet 4.6	✓	✓	✓	✓	7.72 ± 0.32	5.70 ± 0.57	4.92 ± 1.09	5.05 ± 1.00	5.05 ± 0.92
	Opus 4.5	✓	✓	✓	✓	7.73 ± 0.48	5.33 ± 1.03	4.58 ± 1.86	4.33 ± 1.42	4.45 ± 1.51
w/o VSA	Sonnet 4.5	✗	✓	✓	✓	7.17 ± 0.46	4.12 ± 1.09	2.73 ± 1.29	3.02 ± 0.92	2.88 ± 0.94
	Sonnet 4.6	✗	✓	✓	✓	7.55 ± 0.39	5.40 ± 0.72	4.20 ± 1.62	4.35 ± 1.33	4.10 ± 1.28
	Opus 4.5	✗	✓	✓	✓	7.47 ± 0.34	5.42 ± 0.43	3.10 ± 1.58	3.48 ± 0.91	3.33 ± 1.14
w/o Independent Controller	Sonnet 4.5	✓	✗	✓	✓	7.22 ± 0.18	4.80 ± 1.13	2.25 ± 1.01	2.88 ± 0.44	2.38 ± 0.44
	Sonnet 4.6	✓	✗	✓	✓	7.53 ± 0.25	5.53 ± 0.82	4.83 ± 1.72	4.83 ± 1.39	4.70 ± 1.22
	Opus 4.5	✓	✗	✓	✓	7.28 ± 0.25	5.30 ± 0.75	2.92 ± 1.28	3.70 ± 0.78	3.25 ± 1.08
w/o MCP	Sonnet 4.5	✓	✓	✗	✓	7.45 ± 0.42	4.55 ± 0.90	2.40 ± 0.64	3.15 ± 0.39	2.83 ± 0.54
	Sonnet 4.6	✓	✓	✗	✓	7.53 ± 0.30	5.75 ± 0.60	4.12 ± 1.21	4.67 ± 0.95	4.42 ± 0.80
	Opus 4.5	✓	✓	✗	✓	7.08 ± 0.44	4.85 ± 0.64	2.73 ± 0.96	3.33 ± 0.54	3.05 ± 0.73
w/o Reflection	Sonnet 4.5	✓	✓	✓	✗	7.58 ± 0.41	4.25 ± 1.05	3.15 ± 0.95	3.12 ± 0.58	3.08 ± 0.68
	Sonnet 4.6	✓	✓	✓	✗	7.60 ± 0.29	5.88 ± 0.73	4.28 ± 1.26	4.80 ± 0.77	4.42 ± 0.94
	Opus 4.5	✓	✓	✓	✗	7.45 ± 0.61	4.97 ± 0.75	3.58 ± 1.51	3.70 ± 1.06	3.60 ± 1.06

port a contribution from each component but do not rank them against one another.

5 Discussion

5.1 Human validation

The preceding analysis shows that YouRA's mechanisms were exercised in evaluation and that component ablations support their contribution to scalar performance. We next examine how this advantage appears in the failure modes tracked by MLR-Bench: faked experimental results, hallucinated methodology, nonexistent or incorrect citations, and mathematical errors (Chen et al., 2026). For papers generated across the three matched backbones, we first apply the MLR-Judge automated diagnostic using four judges. We report union and intersection counts per category: the former count papers (out of ten per system and backbone) in which at least one judge flagged that category, whereas the latter count only papers flagged by all four judges (Appendix Figure 5). We then conduct a human validation of the automatically flagged instances, summarized in Table 4. Three evaluators independently assess 270 flagged instances (90 per system), confirming roughly three quarters as genuine hallucinations. Precision is highest for hallucinated methodology and lowest for nonexistent citations, and per-system precision is statistically indistinguishable, so the automated diagnostic does not systematically favor or disfavor any system. A fourth evaluator then audits the three annotators by independently re-annotating a 90-instance subset. Agreement is moderate, most disagreements are cases where the auditor is stricter than the primary annotator, and the disagreement rate does not differ significantly across systems ($\chi^2(2) = 2.61$, $p = 0.27$), giving no evidence of systematic bias toward or against any of the three systems.

Because precision is comparable across systems, Table 5 reports precision-corrected hallucination counts. On these corrected counts, MLR-Agent is significantly worse than both other systems, while YouRA and AI Scientist V2 are not statistically distinguishable; this is absence of evidence for a difference, not evidence of equivalence. On Overall, in contrast, YouRA is significantly better than both baselines (§4.2). YouRA thus sits in the low-hallucination group yet alone leads on Overall, so its score advantage is not achieved through more fabricated content.

5.2 Data provenance

The data-provenance companion in Figure 2 and Appendix Table 15 provides a separate provenance context. In Sonnet 4.5, Sonnet 4.6 and Opus 4.5, 27/30 of YouRA's generated papers are classified as real-data-based by both analyzers, compared with 11/30 for MLR-Agent and 17/30 for AI Sci-

Flag precision by category			Flag precision by system			Auditor agreement		
	Count	Rate		Count	Rate		Count	Rate
Hallucinated Methodology	92/102	90.2%	YouRA	68/90	75.6%	YouRA	26/30	86.7%
Faked Experimental Results	64/85	75.3%	MLR-Agent	70/90	77.8%	MLR-Agent	21/30	70.0%
Mathematical Errors	15/21	71.4%	AI Scientist V2	65/90	72.2%	AI Scientist V2	22/30	73.3%
Nonexistent Citations	32/62	51.6%						
Overall	**203/270**	**75.2%**	$\chi^2(2) = 0.75, p = 0.686$			**Overall**	**69/90**	**76.7%**
Wilson 95% CI	[69.7%, 80.0%]					Cohen's κ	0.54	

Table 4: Human validation of the automated hallucination diagnostic. Left: precision of automatically flagged instances by category (confirmed/flagged), with the Wilson 95% CI of the overall precision. Middle: per-system precision; the χ^2 test shows the differences across systems are not significant. Right: agreement between the three primary annotators and an independent fourth auditor on a 90-instance subset (30 per annotator), with Cohen's κ; 18 of the 21 disagreements are cases where the auditor is stricter.

	Flags	Precision	Corrected
YouRA	420	75.6%	317
MLR-Agent	552	77.8%	429
AI Scientist V2	413	72.2%	298
	Δ	95% CI	p
YouRA − MLR-Agent	−112	[−184, −39]	0.003
YouRA − AI Scientist V2	+19	[−45, +86]	0.571
AI Scientist V2 − MLR-Agent	−131	[−200, −62]	<0.001

Table 5: Precision-corrected hallucination counts, computed as automated flags times the human-validated precision from Table 4, with pairwise differences (95% CIs, two-sided p).

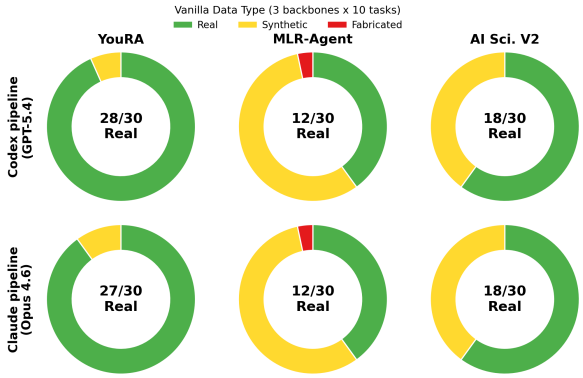


Figure 2: Data-provenance diagnostics on the Sonnet 4.5, Sonnet 4.6, Opus 4.5 tasks subset. Each donut summarizes the Real/Synthetic/Fabricated classification of 30 generated papers for one system under one of two independent analysis pipelines (Codex/GPT-5.4 or Claude/Opus 4.6). Exact counts are in Appendix Table 15.

entist V2. These diagnostics complement the score and preference results in Table 1 and Table 2 with concrete failure-type and data-provenance views.

6 Conclusion

YouRA addresses a persistent gap in autonomous research agents, the drift of manuscript claims away from executed experiments, by making research state, execution evidence, and the research trajectory explicit and durable across the pipeline. VSA-based state externalization, the Independent Controller, and Stateful Reflection record hypotheses, failures, evidence links, and recovery decisions as persistent structures, and removing any one of them significantly lowers end-to-end quality. On MLR-Bench, YouRA improves scalar end-to-end scores over both baselines across matched backbones. Human-validated hallucination counts on the final papers, checked against code and logs, and a provenance classification of the data behind each run's experiments together show that the gain does not rest on fabricated content. YouRA has significantly fewer confirmed hallucinations than MLR-Agent, its count cannot be distinguished from that of AI Scientist V2, its runs rely on real data more often than either baseline under a conservative reading, and the scalar score separates YouRA from both. A scalar end-to-end score cannot by itself tell whether a gain comes from better research or from more persuasive fabrication, so it should be paired with diagnostics that check manuscript claims and their underlying data against executed evidence, such as the human-validated hallucination diagnostic and the automated data-provenance analysis used here.

Limitations

All experiments focus on machine learning and Deep learning tasks, leaving the applicability of YouRA to other scientific domains for future investigation. Research quality is assessed using the MLR-Judge framework with a judge LLM different from those evaluated in the original MLR-Bench study. Alignment with human reviewers therefore requires separate validation for our judge configuration. Input truncation due to the judge LLMs' context length limits may disadvantage papers with extensive code, result files, or logs. The systems compared in our experiments operate with different tool configurations and token budgets, so the reported scores reflect performance under each system's resource conditions. A promising direction for future work is to develop a benchmark with standardized tool access and token budgets, providing a common experimental setting for comparing autonomous research systems under matched resource constraints.

Ethical Considerations

Autonomous paper-generation systems can accelerate legitimate research workflows, but they also create risks of unverified submissions, misleading claims, and over-reliance on AI-generated manuscripts. YouRA is intended to assist researchers, not replace them, so all outputs should be independently verified and AI involvement in manuscript preparation should be disclosed. Because the system uses commercial LLM APIs and external tools, deployments should follow institutional and data-use policies before sending private code, data, or unpublished research ideas to external services; such dependencies may also limit accessibility for under-resourced groups, motivating future support for open-weight or local alternatives. We use MLR-Bench tasks under their stated license and document external-tool usage policies in the released repository. The hallucination and data-provenance diagnostics in this paper are audit tools for checking claim–evidence alignment, not methods for polishing unsupported manuscripts; responsible use requires preserving artifact provenance and making claim-to-evidence links inspectable.

Acknowledgements

This work was supported by Institute of Information and Communications Technology Planning and Evaluation(IITP) grant funded by the Korea government(MSIT) (RS-2019-II190004, Development of semi-supervised learning language intelligence technology and Korean tutoring service for foreigners).

References

Anthropic. 2025a. Claude Opus 4.5 System Card. System card. Released 2025-11-24.

Anthropic. 2025b. Claude Sonnet 4.5 System Card. System card. Released 2025-09-29.

Anthropic. 2026a. Claude Opus 4.6 System Card. System card. Released 2026-02-05.

Anthropic. 2026b. Claude Sonnet 4.6 System Card. System card. Released 2026-02-17.

Tim Baumgärtner and Iryna Gurevych. 2026. SciCoQA: Quality assurance for scientific paper–code alignment. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Hui Chen, Miao Xiong, Yujie Lu, Wei Han, Ailin Deng, Yufei He, Jiaying Wu, Yibo Li, Yue Liu, and Bryan Hooi. 2026. MLR-bench: Evaluating AI agents on open-ended machine learning research. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Mingxuan Du, Benfeng Xu, Chiwei Zhu, Licheng Zhang, Xiaorui Wang, and Zhendong Mao. 2026. Deepresearch bench: A comprehensive benchmark for deep research agents. In *The Fourteenth International Conference on Learning Representations*.

Yolanda Gil, Mark Greaves, James Hendler, and Haym Hirsh. 2014. Amplify scientific discovery with artificial intelligence. *Science*, 346(6206):171–172.

Google DeepMind. 2026. Gemini 3.1 Pro Model Card. Model card. Published 2026-02-19.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, Khaled Saab, Dan Popovici, Jacob Blum, Fan Zhang, Katherine Chou, Avinatan Hassidim, Burak Gokturk, Amin Vahdat, Pushmeet Kohli, and 15 others. 2025. Towards an AI co-scientist. *Preprint*, arXiv:2502.18864.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. 2024. MLAgentBench: evaluating language agents on machine learning experimentation. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Ross D. King, Jem Rowland, Stephen G. Oliver, Michael Young, Wayne Aubrey, Emma Byrne, Maria Liakata, Magdalena Markham, Pinar Pir, Larisa N. Soldatova, Andrew Sparkes, Kenneth E. Whelan, and Amanda Clare. 2009. The automation of science. *Science*, 324(5923):85–89.

Hiroaki Kitano. 2021. Nobel Turing Challenge: creating the engine for scientific discovery. *npj Systems Biology and Applications*, 7(1):29.

Michael Krumdick, Charles Lovering, Varshini Reddy, Seth Ebner, and Chris Tanner. 2026. No free labels: Limitations of LLM-as-a-judge without human grounding. In *Third Conference on Language Modeling*.

Jiachen Liu and Ruihao Zhu. 2026. Agent-native research artifacts. In *The First Conference on Statistics and Trustworthy AI for Cross (X)-Domain Acceleration (Poster Track)*.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. 2024. The AI scientist: Towards fully automated open-ended scientific discovery. *Preprint*, arXiv:2408.06292.

Model Context Protocol Contributors. 2025. Model Context Protocol Specification. Technical specification. Version 2025-03-26.

OpenAI. 2025. Update to GPT-5 System Card: GPT-5.2. System card. Published 2025-12-11.

OpenAI. 2026. GPT-5.4 Thinking System Card. System card. Published 2026-03-05.

Akshara Prabhakar, Roshan Ram, Zixiang Chen, Silvio Savarese, Frank Wang, Caiming Xiong, Huan Wang, and Weiran Yao. 2025. Enterprise deep research: Steerable multi-agent deep research for enterprise analytics. *Preprint*, arXiv:2510.17797.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. 2025. Agent laboratory: Using LLM agents as research assistants. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 5977–6043, Suzhou, China. Association for Computational Linguistics.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Haoyang Su, Renqi Chen, Shixiang Tang, Zhenfei Yin, Xinzhe Zheng, Jinzhe Li, Biqing Qi, Qi Wu, Hui Li, Wanli Ouyang, Philip Torr, Bowen Zhou, and Nanqing Dong. 2025. Many heads are better than one: Improved scientific idea generation by a LLM-based multi-agent system. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28201–28240, Vienna, Austria. Association for Computational Linguistics.

Jiabin Tang, Lianghao Xia, Zhonghang Li, and Chao Huang. 2025. AI-researcher: Autonomous scientific innovation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

xAI. 2026. Grok 4.3. Model documentation. API released 2026-04-30; beta available 2026-04-17; formal email announcement 2026-05-06.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. 2025. The AI scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *Preprint*, arXiv:2504.08066.

Lu Yan, Xuan Chen, and Xiangyu Zhang. 2026. When the specification emerges: Benchmarking faithfulness loss in long-horizon coding agents. *Preprint*, arXiv:2603.17104.

Dingling Zhang, He Zhu, Jincheng Ren, Kangqi Song, Xinran Zhou, Boyu Feng, Shudong Liu, Jiabin Luo, Weihao Xie, Zhaohui Wang, Tianrui Qin, King Zhu, Yuqing Wang, Qianben Chen, Yuchen Eleanor Jiang, Wei Wang, Jiaheng Liu, and Wangchunshu Zhou. 2025. How far are we from genuinely useful deep research agents? *Preprint*, arXiv:2512.01948.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23.

Qiran Zou, Hou Hei Lam, Wenhao Zhao, Yiming Tang, Tingting Chen, Samson Yu, Tianyi Zhang, Chang Liu, Xiangyang Ji, and Dianbo Liu. 2026. Fml-bench: Benchmarking machine learning agents for scientific research. *Preprint*, arXiv:2510.10472.

A System Implementation Details

This appendix explains how the VSA, Independent Controller, and Stateful Reflection mechanisms introduced in the main methodology are implemented in the deployed workflow. The implementation is organized into four layers: external tool integration, state externalization mechanisms for long-horizon persistence, a centralized state file shared by all agents, and stage-specific agent assignments.

External Tool Integration. YouRA uses Claude Code as the execution host and connects the five MCP-facing tools in Table 6 as a common tool-access layer. **Archon** provides retrieval-augmented generation (RAG)-backed knowledge management, storing project-specific documents and prior implementation cases in a vector database. **Exa** offers Search, Contents, and Research APIs for web-based evidence collection. **Semantic Scholar** supplies Academic Graph and Recommendations APIs for paper metadata and citation networks. **Serena** enables symbol-level code navigation and editing,

with failure-context recording for Reflective Memory. **Clear Thought** provides structured reasoning scaffolds, including scientific-method and mental-model templates. Claude Code hooks provide termination control and iterative execution loops; Table 6 lists only the external tool services exposed through MCP-style access.

State Externalization Mechanisms. MCP tool calls are stateless, so YouRA separates transient tool calls from durable artifacts. BMAD-Method[3] provides structured artifact templates so each stage receives the prior stage's outputs in a fixed schema. During Problem Scoping, YouRA adapts BMAD's brainstorming workflow as a Socratic Discovery procedure for topic expansion: a topic-level prompt is expanded into diverse research angles through iterative questioning, anti-bias pivots, and convergence into a research-question package passed to Literature Grounding. Archon's RAG-backed knowledge management stores project-specific documents, code patterns, and prior implementation cases in a vector database, which agents query during Implementation Planning and Coding & Validation. Agent Skills[4] loads procedural knowledge on demand, including prompt templates, verification protocols, and output schemas. These mechanisms are supporting infrastructure around the VSA: they keep state available outside the transient conversation context.

Verification State Architecture. The VSA invariants are introduced in §3.2. Every agent reads from and writes to the centralized YAML file, and Figure 3 shows a simplified schema. The schema excerpt preserves implementation labels used for machine routing; the surrounding prose uses reader-facing stage names.

Stage-Agent Assignment. Table 7 summarizes which agents and tools the controller invokes from each VSA stage state. Agents fall into three functional groups. *Planning agents* span Problem Scoping through Experiment Design, turning interests into grounded questions, debated hypotheses, verification plans, and experiment specifications. *Execution agents* cover Implementation Planning and Coding & Validation, producing PRDs, architectures, task lists, implementations, and validation outcomes. *Synthesis agents* cover Evidence Check, Manuscript Drafting, Adversarial Fact Review, and Manuscript Refinement, carrying executed evidence into manuscript claims and review. The Tools column summarizes external tool access; all stages read from and write to the VSA.

```yaml
metadata:
project_name: "..."
main_hypothesis_id: H-main
schema_version: "3.5"

workflow:
status: ACTIVE
current_phase: Phase 4.5
next_action: "..."
failure_routing:
  phase4_must_work_partial:
    route_after_max: Phase 2A-Dialogue
  phase4_must_work_fail:
    route_to: Phase 0

sub_hypotheses:
h-e1:
  status: COMPLETED
  gate: {type: MUST_WORK, result: PASS}
  experiment_design:
    file: h-e1/02c_experiment_brief.md
  implementation_planning:
    tasks_file: h-e1/03_tasks.yaml
  validation:
    result: PASS
    key_findings: ["..."]

history:
- event: Phase 4 PASS - h-e1 validated
timestamp: "..."

statistics:
validated_sub_hypotheses: 3
gates_passed: 3
```

Figure 3: Simplified excerpt of the VSA state file. `workflow` governs stage transitions and recovery routing; each `sub_hypotheses` entry stores gate status, artifact pointers, and validation findings; `history` records auditable state transitions. Implementation labels are shown as stored in the machine-readable state, while the prose uses reader-facing stage names.

Independent Controller. The Independent Controller turns VSA records into stage transitions, recovery choices, and persona selection, making persistent state an active control surface rather than a passive log. It runs in an independent LLM context decoupled from the task-execution agent; in our implementation, this controller is GPT-5.2 (OpenAI, 2025), accessed through OpenRouter API. At each step it reads only two inputs: a compact summary of the current VSA and the most recent per-stage artifact pointer. Because the controller never sees the full execution trace, its context does not grow with the pipeline length, and its routing decisions are less directly exposed to the task-execution agent's internal reasoning. The same Independent Controller is reused in Hypothesis Debate, where it selects personas and judges convergence; in Adversarial Fact Review, the independent critics are the reviewer personas, which run as context-isolated task agents of the backbone LLM, while the con-

[3] https://github.com/bmad-code-org/bmad-method
[4] https://agentskills.io/specification

Tool	Role	Lifecycle coverage	Usage in YouRA
Archon	Sequential Memory	Full lifecycle	RAG-backed knowledge base; stores implementation patterns; manages task lifecycle (todo/doing/review/done)
Exa	Evidence Search	Grounding, debate, planning, validation	Web-based evidence collection via Search, Contents, and Research APIs
Semantic Scholar	Literature Search	Grounding, debate, evidence, writing, review	Academic Graph and Recommendations APIs for paper metadata and citations
Serena	Code Analysis / Memory	Scoping, debate, hypothesis loop, writing, review	Code-aware symbol navigation; optional in Experiment Design when base code exists, stronger in Implementation Planning–Coding & Validation; memory support where explicitly invoked
Clear Thought	Structured Reasoning	Debate and evidence checking	Scientific method and mental models for hypothesis refinement

Table 6: MCP-facing tool-access stack used by YouRA. Claude Code is the execution host and hook surface and has no row of its own; the rows list external tool services exposed through MCP-style access. Tool URLs: Archon (https://github.com/coleam00/Archon/tree/archive/v1-task-management-rag), Exa (https://exa.ai), Semantic Scholar (https://www.semanticscholar.org/product/api), Serena (https://github.com/oraios/serena), Clear Thought (https://github.com/Dicklesworthstone/mcp-clear-thought).

troller's Stop hook handles only round-to-round continuation. Operationally, the controller's per-stage output is consumed by a Stop hook that either approves termination, emits a resume prompt back to the task-execution LLM, or escalates a `MUST_STOP` signal when human intervention is required.

Reflection-Guided Failure Handling. When a Coding & Validation gate fails, recovery first separates local repair from archive-producing escalation. Local runtime, test, or fabricated-output failures are handled as bounded repair inside Coding & Validation; such attempts may appear in validator traces but do not create separate archive snapshots. Only when the gate remains unsatisfied does the reflection step classify the failure by recoverability and evidence contradiction, then select same-hypothesis retry, Hypothesis Debate redesign, or Problem Scoping reset. Table 8 summarizes these externally visible routing levels, and Algorithm 1 gives the paper-level decision logic. Reflective Memory and the VSA `recovery_history` record routing-relevant failure context and escalated outcomes, so downstream stages inherit constraints learned from failure. Sequential Memory (Archon) records the archive-producing routing/recovery events summarized in Table 9.

Hypothesis Debate. Hypothesis Debate implements the function of multi-persona scientific debate inside YouRA's stateful workflow. The six personas cover novelty and ideation (Dr. Nova), falsifiability and experiment design (Prof. Vera), scientific contribution and impact (Dr. Sage), technical/theoretical feasibility (Prof. Pax), evidence-based hypothesis refinement (Dr. Ally), and assumptions/failure modes (Prof. Rex). The protocol has three stages. First, Gap Selection chooses the priority research gap and supporting evidence from Literature Grounding. During Debate, the independent controller LLM (GPT-5.2 (OpenAI, 2025)) reads recent dialogue and VSA state to decide convergence and select the next personas, while the backbone LLM produces the role-conditioned responses. If a network or API failure prevents a controller call, backbone-LLM fallback preserves execution continuity for that round. Result Structuring converts the debate outcome into structured YAML for downstream stages. When this stage

Workflow stage	Tools	Input	Output
Problem Scoping	Archon, Serena	Researcher's interests	Concrete research question
Literature Grounding	Archon, SS, Exa	Research question	3 research gaps
Hypothesis Debate	Archon, SS, Exa, Serena, ClearThought	3 research gaps	Formalized hypothesis
Verification Planning	Archon, Exa, ClearThought	Formalized hypothesis	3–7 sub-hypothesis DAG
Experiment Design	Archon, Exa, Serena	Sub-hypotheses	Experiment specification
Implementation Planning	Archon, Serena	Experiment spec	PRD, architecture, logic, config, tasks
Coding & Validation	Archon, Exa, Serena	Tasks	Validated implementation
Evidence Check	Archon, Serena	All Coding & Validation artifacts	Evidence-check summary
Manuscript Drafting	Archon, Serena	Evidence-check summary	Draft manuscript
Adversarial Fact Review	Archon, Serena	Manuscript + VSA + code	Reviewed paper
Manuscript Refinement	–	All artifacts	Refined final paper

Table 7: Stage-agent assignment. **SS** = Semantic Scholar. The Tools column lists external tool access; all stages read from and write to the VSA.

Level	Target	Criterion
Repair	Coding & Validation (in-loop)	Local runtime, test, or fabricated-output failure repairable before archive-producing escalation
Redesign	Hypothesis Debate	`MUST_WORK` failure requiring hypothesis or mechanism revision after repair
Reset	Problem Scoping	Evidence contradiction indicating that the current hypothesis is unsupported

Table 8: Reflection-guided failure-routing levels. Bounded repair is handled in-loop and does not create an archive snapshot; redesign and reset are archive-producing escalations.

is re-entered after a failed or superseded hypothesis, Reflective Memory is checked first so the system does not repeat refuted directions. The converged hypothesis is formalized with variables, causal mechanisms, a null hypothesis, and testable predictions, then decomposed during Verification Planning into a 3–7-node sub-hypothesis DAG with `MUST_WORK/SHOULD_WORK` gates recorded in the VSA.

Hypothesis Loop Details: Experiment Design through Coding & Validation. Each sub-hypothesis traverses the hypothesis loop from Experiment Design through Coding & Validation. Experiment Design turns the sub-hypothesis into an experiment specification covering the dataset, model architecture, and evaluation metrics. Implementation Planning expands this specification into PRD, architecture, logic, and configuration documents, then consolidates them into the VSA task artifact `03_tasks.yaml`, a prioritized implementation task list whose entries store references to the relevant specification sections. Coding & Validation uses these references to apply SDD (Spec-Driven Development): for each task, it extracts the governing requirements, generates and runs compliance tests before implementation, implements the task, and invokes an independent Validator for multi-stage verification. Failures are fed back with diagnostic context, unmet gates trigger reflection-guided routing, and a hook-based mock-data/hard-coded-result detector checks whether outputs come from actual data processing or from fabricated placeholders. Each completed sub-hypothesis is marked as validated, refuted, or escalated; the loop continues until all sub-hypotheses reach a terminal state, after which the pipeline proceeds to Evidence Check.

Evidence Check. Evidence Check produces an evidence-check summary that records, for each prediction, whether the executed result supports it (SUPPORTED, PARTIALLY_SUPPORTED, REFUTED, or INCONCLUSIVE); the same pass also

Algorithm 1 Reflection-guided failure handling in the hypothesis loop

Require: Failed gate type g, failure context c, retry state r
Ensure: Next action a

if the failure is local and a bounded Coding & Validation repair attempt remains **then**
  return bounded repair in Coding & Validation
end if
if $g =$ `SHOULD_WORK` **then**
  if a same-hypothesis improvement path remains available **then**
    SAME-HYPOTHESIS RETRY decision
    return retry from Experiment Design within the current hypothesis
  else
    record the limitation in Reflective Memory and the VSA
    return continue the pipeline with the limitation documented
  end if
end if
$\alpha \leftarrow$ reflection over failure severity, recoverability, and evidence contradiction
write α to Reflective Memory and the VSA `recovery_history`
if $\alpha =$ SAME-HYPOTHESIS RETRY **then**
  return retry from Hypothesis-Loop within the current hypothesis
else if $\alpha =$ REDESIGN **then**
  return reroute to Hypothesis Debate for hypothesis redesign
else
  return reroute to Problem Scoping for question reset
end if

tracks explicit causal-mechanism claims and assumptions. This stage is an evidence-transfer check: it records which predictions and assumptions match executed results, then passes that summary to Manuscript Drafting and Adversarial Fact Review. The resulting evidence-check summary is a core input to writing and review alongside the VSA and Coding & Validation artifacts; its downstream effect on manuscript-claim traceability is assessed through the hallucination and data-provenance diagnostics in Appendices E and F.

Manuscript Drafting: Story Group Architecture. Manuscript Drafting generates the manuscript in three sequential story groups, guided by a blueprint that is written *before* any section content. *Foundation* (Introduction + Related Work + Methodology) frames the problem and motivates the solution. *Evidence* (Experiments + Results + Discussion) presents and interprets the experimental findings. *Closure* (Conclusion + Abstract) is produced last, and the abstract is written last to capture the full paper context. The Evidence Check summary is one input to manuscript writing, alongside the VSA and artifacts from Problem Scoping through Coding & Validation (e.g., Problem Scoping for introduction motivation, Experiment Design/Implementation Planning for methodology, Coding & Validation for results). VSA-recorded values are the intended source for reported numbers, discouraging unsupported reported values.

Adversarial Fact Review and Manuscript Refinement. Adversarial Fact Review applies an adversarial fact-review protocol (*Devil's Advocate*) with three reviewer personas. The *Accuracy Checker* cross-references every claim against ground truth, compares reported numbers against a dedicated ground-truth registry, and checks methodology consistency and cross-section contradictions against Coding & Validation artifacts. The *Bored Reviewer* examines whether the abstract and framing would compel further reading from the perspective of a time-pressed conference reviewer. The *Skeptical Expert* probes novelty claims and baseline fairness. Findings are classified into two blocking levels. *Major* findings are claim–implementation contradictions or reviewer-attackable weaknesses such as unfair baselines that must be corrected with evidence before convergence; *Minor* findings are polish issues such as typos, awkward phrasing, and formatting, collected as human-review notes. Manuscript Refinement revisits the VSA, Evidence Check summary, stage artifacts, result files, logs, and code until no major findings remain.

B Reproducibility Details

This section summarizes the code/artifacts, failure-routing logic, and execution configuration needed to reproduce the core results.

Code and Artifacts. The complete source code, VSA schemas, controller configurations, agent prompts, evaluation scripts, generated papers, VSA state files, recovery snapshots, judge prompts and raw outputs, hallucination-diagnostic outputs, and table/figure regeneration scripts are released at `https://github.com/PrayPrey/Your-Research-Agent`.

Dataset and Artifact Licensing. MLR-Bench tasks (Chen et al., 2026) are released by the original authors under their stated license, and we use them for non-commercial academic evaluation only. The reference papers, datasets, and starter code that each task points to are accessed under the licenses of their original sources and are not redistributed by

us. Manuscripts and code generated by YouRA in the course of these benchmark runs are released for research use under the same terms as the YouRA repository and inherit any upstream licensing obligations from the underlying datasets they exercise; users who repurpose generated artifacts must comply with those upstream terms.

Failure Classification Logic. The Validator and reflection step together map each gate failure to one of the three externally visible routing levels in Table 8; Algorithm 1 gives the compact paper-level decision procedure. Repair is bounded in-loop handling inside Coding & Validation, whereas redesign and reset are escalations that write structured archive snapshots.

- **Repair** (runtime / test / fabricated-output failure): the implementation produces an exception, assertion error, test failure, or mock-data/hard-coded-output signal that is localizable to a specific code location. The Validator attempts bounded in-place fixes within Coding & Validation before any archive-producing escalation.
- **Redesign routing** (design / mechanism failure): a `MUST_WORK` failure persists after bounded repair, but reflection judges that the failure remains recoverable through hypothesis or mechanism revision (e.g., architectural mismatch, invalid causal mechanism, dependency structure that requires redesign). The system routes to Hypothesis Debate with a diagnostic record specifying the failed component and attempted fixes.
- **Reset** (fundamental failure): a `MUST_WORK` failure reflects evidence that fundamentally contradicts the current hypothesis (e.g., predicted correlation does not hold, effect size remains absent after correct implementation). The system routes to Problem Scoping with the full failure context, archiving the current research folder under a timestamped backup before beginning a new research direction.

Execution Configuration. All experiments use the backbone LLM's default sampling parameters (temperature and top-p as set by the provider's API defaults). We evaluate Coding & Validation reflection-guided routing by reporting observed archive-producing routing/recovery counts for each ten-task backbone batch in Table 9 and Appendix Tables 10–11. The six-persona debate in Hypothesis Debate runs for up to 20 rounds in our implementation (one persona contribution per round), terminating when all six convergence criteria (specificity, mechanism clarity, testable predictions, novelty, feasibility, and objection resolution) are jointly satisfied. Coding & Validation SDD cycles iterate until gate criteria resolve through validation, refutation, or escalation. The MCP tool stack is listed in Table 6; each server is installed from its upstream repository.

Reproducibility Checklist Disclosure. We disclose the reproducibility-relevant configuration for this version: (i) **random seeds** are not used as a reproducibility guarantee for the controller/evaluation layer because it routes through multiple commercial LLM API paths (OpenRouter / Anthropic and provider-backed judge endpoints) that do not provide a uniform deterministic replay guarantee across models and backend updates; we use the provider-default API sampling parameters stated above. (ii) **Hardware specification.** All experiments ran on a single workstation with the following configuration: 2× AMD EPYC 9354 32-core CPUs (total 64 physical cores / 128 threads), 128 GB system RAM, up to 5× NVIDIA H100 NVL GPUs (94 GB HBM3 each; NVIDIA driver 575.57.08), Ubuntu 22.04 LTS (Linux kernel 5.15), and Python 3.12. The controller/evaluation layer (Hypothesis Debate controller, judge calls, baseline LLM backbones) runs entirely against commercial APIs and consumes no local GPU. The Coding & Validation *execution* layer, in contrast, runs the LLM-generated PyTorch experiment code locally on the workstation's GPU(s); the actual per-task GPU count varies with the experiment generated by the agent (typically 1 GPU; multi-GPU is only used when the generated code explicitly requests it). All experiments are single-node and use no multi-node distributed training.

C Operational Telemetry

This appendix reports operational statistics for the three-level routing mechanism (§3.4), aggregated from the `verification_state.yaml` final state and routing-recovery archive snapshots of the benchmark artifacts (10 MLR-Bench tasks × 3 backbones). Aggregation, classification, and end-to-end verification scripts are released with the repository; every value reported here was independently re-derived from raw artifacts as a 30/30 task

Backbone	Tasks	Total	Per task	Median	Range
Sonnet 4.5	10	28	2.8	2.5	0–8
Opus 4.5	10	40	4.0	3.0	0–11
Sonnet 4.6	10	18	1.8	1.5	0–4

Table 9: Observed archive-producing reflection/routing events by backbone. Each event corresponds to one escalation snapshot generated after a Coding & Validation gate failure survives bounded in-loop repair; bounded repair attempts themselves are excluded by design. *Per task* and *Median* summarize the per-task archive-producing event count.

Task(MLR-Bench)	Sonnet 4.5	Opus 4.5	Sonnet 4.6
bi_align	2	3	2
buildingtrust	0	7	1
data_problems	3	3	0
dl4c	8	4	1
mldpr	3	6	1
question	7	11	4
scope	0	3	2
scsl	4	3	3
verifai	1	0	3
wsl	0	0	1
Total	**28**	**40**	**18**

Table 10: Routing event count per MLR-Bench e2e benchmark task and backbone. The 10 tasks cover selected MLR-Bench research topics, from alignment and trust to data, code, uncertainty, optimization, verification, and neural-network weights. Counts are computed from archived routing-recovery snapshots within each task's working directory.

verification check before submission. Table 9 provides the compact backbone-wise summary, and the subsequent tables give per-task and per-level evidence.

Routing event counts (Table 10). Each routing-recovery event creates a timestamped archived snapshot of the VSA. Counts therefore equal the number of times the pipeline left Coding & Validation to redesign or reset within a task. Each (backbone, task) cell uses the final working directory of that task only.

Routing levels (Table 11). Each archive snapshot is classified into one of the three Table 8 routing levels using a cascade of signals (`episode.routing_decision` → `termination_trigger` + `routing_reason` keywords → `failures_by_type` counts → `history` fallback). Snapshots without any parseable termination signal are reported as *unclassified* and not discarded.

Backbone	redesign	reset	unclass.	Total
Sonnet 4.5	3	25	0	28
Opus 4.5	1	39	0	40
Sonnet 4.6	1	17	0	18
Total	**5**	**81**	**0**	**86**

Table 11: Routing events by classified level *among archive-producing events*, computed over the same final-retry-only working directories as Table 10 (row totals match cell-wise). Bounded repair runs frequently in-loop within Coding & Validation and produces no archive snapshot by design; this distribution therefore describes escalated failures only.

Tables 10–11 together show 86 archive-producing reflection/routing events across the reported final-directory cells: 28 for Sonnet 4.5, 40 for Opus 4.5, and 18 for Sonnet 4.6. The compact event-count summary appears in Table 9.

Reflective Memory record format. Each routing event writes a structured record to Serena under the file pattern `failure_{hypothesis_id}_run{n}.md` (or `pivot_{old}_{new}.md` for redesign-induced supersession). Subsequent Hypothesis Debate discussions consult these records before generating new candidates, using refuted approaches as constraints and partially validated elements as reusable evidence. Figure 4 shows an anonymized record drawn from one of the 30 benchmark runs.

D Supplementary Interpretation Details

This appendix clarifies supplementary context behind the main results: tool-access differences, observed recovery-routing distribution and the 201-task upstream Idea/Proposal analysis. The automated diagnostic based on MLR-Bench's hallucination taxonomy is reported separately in Appendix E, and order-bias-controlled pairwise preferences remain in the main text as Table 2 because they are part of the primary result.

D.1 Statistical Significance of the End-to-End Comparison

The unit of analysis is the matched (backbone, task) cell. Each cell score is the mean of the four LLM judges' Overall scores, and because the three systems share the same 30 cells, task and backbone variation cancels within pairs. We test the mean Overall difference with a paired sign-flip permuta-

```
# Phase 4 Failure Record: h-e2 (Run 2)

Date: 2026-02-14T03:21:08
Hypothesis: h-e2
Final Status: FAIL
Failure Type: HYPOTHESIS_ISSUE

## Performance Gap
| Metric | Ours | Baseline | Gap |
|---------|------|----------|---------|
| WGA (%) | 71.3 | 78.6 | -7.3 |

## Root Causes
- CV-ratio threshold collapses on
  imbalanced minority groups
- Predicted monotonic separation does
  not hold under label noise > 5%

## Lessons Learned
- Threshold-based separation needs a
  robustness term for noisy labels
- Worst-group accuracy regresses when
  group size < 200

## Routing Decision
redesign -> Phase 2A (mechanism
revision: replace fixed threshold
with adaptive quantile)
```

Figure 4: Anonymized Reflective Memory record (Coding & Validation failure, redesign route). Implementation labels are shown as recorded in the raw memory dump; the corresponding prose stage names are Coding & Validation and Hypothesis Debate. The identifier `h-e#` indexes sub-hypotheses within a parent hypothesis, and "Run 2" marks the second routing-recovery snapshot. The unedited Reflective Memory dumps that each benchmark run produced ship with the released repository.

tion test using 100,000 random sign assignments (two-sided). Table 12 summarizes the results: YouRA leads MLR-Agent by $+0.85$ (p=0.006) and AI Scientist V2 by $+0.64$ (p=0.044), with both comparisons still significant under Holm correction (adjusted p=0.012 and 0.044). The per-backbone mean differences are positive across all three backbones for both comparisons, so no single backbone drives the result. Cells sharing a backbone are not fully independent; the permutation runs at the cell level, not on individual judge verdicts, which avoids treating the 120 judge scores as independent samples.

D.2 201-Task Upstream Context

Table 13 compares Idea/Proposal-stage quality between YouRA and MLR-Agent on the full MLR-Bench task pool. Both systems were run with the Sonnet 4.5 and Opus 4.5 backbones, and a single Gemini 3.0 Pro judge scored both systems' outputs under the MLR-Bench stage rubrics. A Claude Opus 4.5 adapter at temperature 0 converted YouRA's hypothesis artifacts to the MLR-Bench idea/proposal format. Table 14 reports task-paired sign-flip permutation tests. On Sonnet 4.5, MLR-Agent scores significantly higher on Overall at the Idea stage and on Clarity and Soundness at the Proposal stage; the Overall gap there does not survive correction. On Opus 4.5, Overall does not differ, with YouRA higher on Novelty and Feasibility and MLR-Agent higher on Soundness and Significance. This upstream comparison contrasts with the end-to-end paper-level comparison in Table 1, and it excludes AI Scientist V2 because that system was not run in this setting.

Table 12: Paired significance tests for end-to-end Overall comparisons. The unit is the 30 matched (backbone, task) cells, where each cell averages four judges' Overall scores. Permutation p is two-sided from 100,000 sign-flip assignments. W/L/T reports cell-level wins, losses, and ties for YouRA. Sign p is the exact two-sided sign-test p after excluding ties. Backbone deltas are reported in Sonnet 4.5 / Opus 4.5 / Sonnet 4.6 order.

Comparison	Δ Overall	Perm. p	Holm p	W/L/T	Sign p	Backbone deltas
YouRA − MLR-Agent	$+0.85$	0.006	0.012	18/12/0	0.362	$+1.10/+0.83/+0.63$
YouRA − AI Scientist V2	$+0.64$	0.044	0.044	20/8/2	0.036	$+0.58/+0.18/+1.18$

E Automated Hallucination Diagnostic

Protocol. MLR-Bench's hallucination analysis defines four fact-based failure types (nonexistent or incorrect citations, hallucinated methodology, mathematical errors, and faked experimental results) and applies human verification after automated LLM detection (Chen et al., 2026). We apply the automated detection stage to the generated papers across all three matched backbones, updating the original Gemini/Claude two-judge setup to a four-judge panel: Grok 4.3, GPT 5.4, Gemini 3.1 Pro, and Claude Opus 4.6. We report the results as judge-union and judge-intersection bounds, and validate the automatically flagged instances with three human annotators and an independent auditor (Table 4; §5.1).

Union/intersection semantics. Union counts capture any judge-flagged case, while intersection counts capture only cases flagged by all four judges. Figure 5 shows intersection counts in the top row and union counts in the bottom row; bar labels give the number of flagged papers out of ten. The gap between the two values reflects judge disagreement, and we report both instead of reducing the diagnostic to a single hallucination score.

Stage	System	Backbone LLM	Consistency	Clarity	Novelty	Soundness	Feasibility	Significance	Overall
Idea	YouRA	Sonnet 4.5	9.49 ± 1.65	8.38 ± 0.84	7.32 ± 0.88	–	6.75 ± 1.44	8.25 ± 1.10	7.90 ± 1.37
		Opus 4.5	9.91 ± 0.60	8.65 ± 0.69	$\mathbf{7.49 \pm 0.76}$	–	$\mathbf{7.32 \pm 1.27}$	8.31 ± 0.71	8.40 ± 0.83
	MLR-Agent	Sonnet 4.5	$\mathbf{9.97 \pm 0.30}$	$\mathbf{8.62 \pm 0.67}$	7.19 ± 0.71	–	6.76 ± 1.03	$\mathbf{8.46 \pm 0.55}$	$\mathbf{8.25 \pm 0.64}$
		Opus 4.5	9.98 ± 0.17	8.69 ± 0.60	7.24 ± 0.68	–	6.98 ± 1.07	$\mathbf{8.49 \pm 0.67}$	8.34 ± 0.67
Proposal	YouRA	Sonnet 4.5	9.74 ± 0.87	8.86 ± 0.87	7.89 ± 0.85	6.78 ± 1.73	7.00 ± 1.55	8.24 ± 1.29	7.79 ± 1.66
		Opus 4.5	$\mathbf{9.92 \pm 0.28}$	9.09 ± 0.68	$\mathbf{7.93 \pm 0.76}$	6.96 ± 1.64	$\mathbf{7.58 \pm 1.46}$	8.17 ± 1.04	7.95 ± 1.46
	MLR-Agent	Sonnet 4.5	9.73 ± 0.95	$\mathbf{9.12 \pm 0.81}$	7.72 ± 1.25	$\mathbf{7.51 \pm 1.28}$	6.89 ± 1.07	8.48 ± 0.94	8.14 ± 1.35
		Opus 4.5	9.62 ± 1.32	9.11 ± 0.56	7.47 ± 1.49	$\mathbf{7.42 \pm 1.40}$	7.14 ± 1.21	8.33 ± 1.26	8.06 ± 1.56

Table 13: Idea/Proposal-stage scores of YouRA and MLR-Agent on the MLR-Bench task pool. Both systems' outputs are scored by the same single Gemini 3.0 Pro judge under the MLR-Bench stage rubrics (1–10, higher better); YouRA's hypotheses are converted to the MLR-Bench idea/proposal format by a Claude Opus 4.5 adapter at temperature 0. "–" = metric not defined. Bold marks the higher mean where the paired difference is significant after Holm correction (Table 14).

Table 14: Paired tests for the Idea/Proposal-stage scores in Table 13 (YouRA minus MLR-Agent, matched by task). Δ Overall is the mean paired difference, Perm. p is a two-sided sign-flip permutation p (20,000 assignments), and Holm p is corrected over the rubric dimensions within each stage–backbone block. The last two columns list the dimensions on which each system is significantly higher after correction.

Stage	Backbone	Δ Overall	Perm. p	Holm p	YouRA higher	MLR-Agent higher
Idea	Sonnet 4.5	-0.35	0.001	0.004	–	Cons., Clar., Sig., Overall
Idea	Opus 4.5	$+0.08$	0.246	0.739	Nov., Feas.	Sig.
Proposal	Sonnet 4.5	-0.36	0.020	0.099	–	Clar., Sound.
Proposal	Opus 4.5	-0.11	0.428	0.857	Cons., Nov., Feas.	Sound.

Interpretive takeaway. Union counts are high for most rows, so we report the diagnostic as an automated profile of hallucination flags, and keep union and intersection counts separate; the profile is not used as a standalone ranking metric. On the all-judge-agreed intersection flags, YouRA shows a favorable profile on execution-evidence categories such as hallucinated methodology and faked experimental results; we report these as hallucination-flag counts; the human validation in §5.1 (Table 4) estimates the precision of these automated flags at 75.2% overall. Citation flags vary by backbone and are treated as a secondary signal. This pattern complements the Soundness/Overall gains in Table 1 and the real-data provenance contrast in Figure 2 with concrete failure-type and data-provenance views.

F Data Provenance Analysis

Protocol. Each generated paper is classified at the paper level by the kind of data its experiments ran on. An automated analyzer receives the paper and the root of its experiment folder, explores the folder without assuming its layout, and inspects the experiment code and result files to check that execution actually occurred, then assigns one of three labels: *Real* (experiments run on real datasets, e.g., loaded through `datasets.load_dataset` or `torchvision.datasets`), *Synthetic* (experiments run on artificially generated data, e.g., random draws or synthetic-dataset generators), or *Fabricated* (reported values without runnable experiment code, e.g., hardcoded result arrays). Every label is stored with the file-level evidence it rests on. Two analyzers built on distinct model families, Claude Code (Opus 4.6) and Codex (GPT-5.4), run the same protocol independently, so the classification does not hinge on a single model family; they agree on 85 of the 90 (system, backbone, task) cells. The counts reported in §5.2 use a conservative, maximum-synthetic reading: a run counts as real-data-based only when both analyzers label it *Real*, so a *Synthetic* or *Fabricated* label from either analyzer counts against the run. Table 15 breaks the counts down by backbone and analyzer.

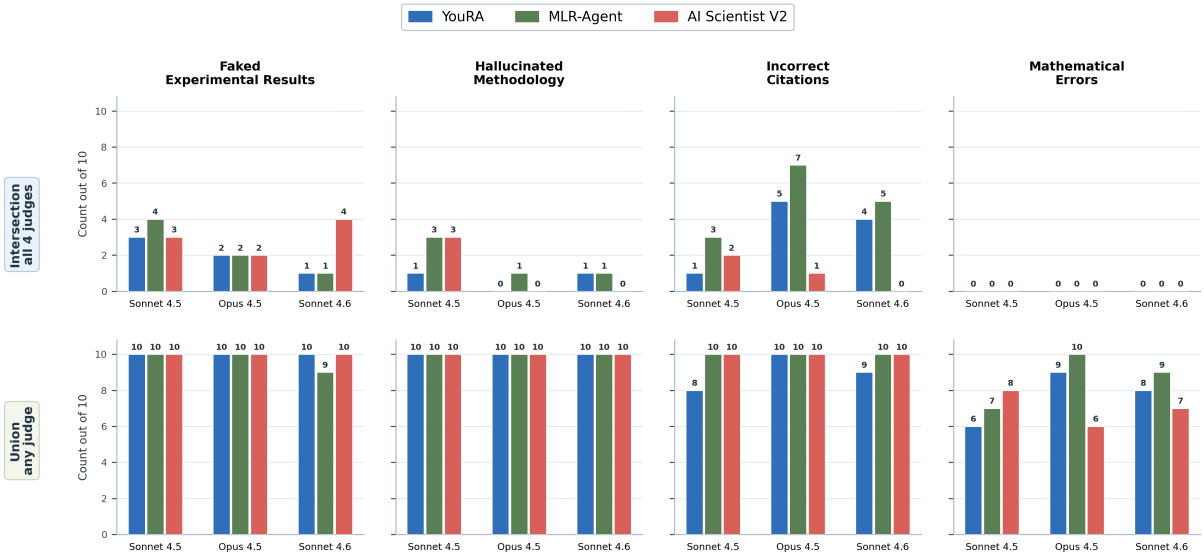


Figure 5: Automated MLR-Bench hallucination diagnostic across matched backbones. The top row reports all-judge intersection counts out of ten papers, requiring all four judges to identify the hallucination type, while the bottom row reports any-judge union counts, requiring at least one judge to identify it. The four category columns denote nonexistent/incorrect citations (Cite), hallucinated methodology (Method), mathematical errors (Math), and faked experimental results (Faked). The diagnostic adapts MLR-Bench's automated hallucination categories but does not include the human verification step used in the original MLR-Bench hallucination analysis.

Table 15: Real-data classifications per system and backbone (out of ten tasks): the conservative both-analyzer count used in the main text, and each analyzer separately.

System	Backbone	Both analyzers	Opus 4.6	GPT-5.4
YouRA	Sonnet 4.5	8/10	8/10	9/10
	Opus 4.5	9/10	9/10	9/10
	Sonnet 4.6	10/10	10/10	10/10
AI Scientist V2	Sonnet 4.5	6/10	6/10	6/10
	Opus 4.5	8/10	9/10	8/10
	Sonnet 4.6	3/10	3/10	4/10
MLR-Agent	Sonnet 4.5	1/10	2/10	2/10
	Opus 4.5	4/10	4/10	4/10
	Sonnet 4.6	6/10	6/10	6/10

Counts by analyzer. Each analyzer taken alone gives the same ordering: 27/30, 12/30, and 18/30 real-data runs for YouRA, MLR-Agent, and AI Scientist V2 under the Opus 4.6 analyzer, and 28/30, 12/30, and 18/30 under GPT-5.4. All non-real cells are Synthetic except one MLR-Agent case labeled Fabricated. The analyzers assign labels automatically and record per-case evidence so that each classification can be inspected.

G Component Ablations

This appendix interprets the ablation rows in main Table 3 and reports their statistical tests. All four component ablations use Sonnet 4.5, Opus 4.5, and Sonnet 4.6. Each one removes a single component from full YouRA while leaving the rest of the workflow in place: *w/o MCP* removes the MCP tool layer, *w/o Reflection* disables reflection-guided redesign and reset, *w/o VSA* removes the persistent state record that the model reads and updates, and *w/o Independent Controller* removes the separate high-level controller. The analysis unit is the matched (backbone, task) cell. Each cell score averages the four judges' Overall scores, so every component ablation has 30 matched cells (three backbones $\times$ ten tasks).

Ablation significance. Table 16 compares full YouRA with each ablated configuration over the same 30 cells. Removing any of the four components lowers *Overall*, and all four differences stay significant after Holm correction under paired sign-flip permutation tests. These tests do not rank the components.

Table 16: Paired significance tests for component removals. The unit is the 30 matched (backbone, task) cells, where each cell averages four judges' Overall scores. Δ Overall is ablated minus full YouRA, so negative values indicate drops under ablation. Permutation p is two-sided from 100,000 sign-flip assignments. W/L/T reports cells where full YouRA is higher, lower, or tied. Backbone drops are reported in Sonnet 4.5 / Opus 4.5 / Sonnet 4.6 order.

Removed component	Δ Overall	Perm. p	Holm p	W/L/T	Sign p	Backbone drops
MCP	-1.13	< 0.001	< 0.001	20/5/5	0.004	$-1.38/-1.40/-0.63$
Reflection	-0.87	0.007	0.007	17/11/2	0.345	$-1.13/-0.85/-0.63$
VSA	-1.13	0.002	0.004	22/8/0	0.016	$-1.32/-1.12/-0.95$
Independent Controller	-1.12	0.001	0.003	22/6/2	0.004	$-1.82/-1.20/-0.35$

Table 17 compares these drop sizes directly across removals. Every pairwise contrast has Holm-adjusted permutation p=1.000, so this sample cannot support a reliable ranking of components by drop size. The two analyses show that each component contributes while leaving their relative importance open.

Table 17: Paired contrasts between component drop sizes. Each row tests the difference between two removal drops over the same 30 matched cells. Positive values mean that the first removal has the larger drop. W/L/T reports cells where the first drop is larger, smaller, or tied.

Drop-size contrast	Δ drop	Perm. p	Holm p	W/L/T
MCP $-$ Reflection	$+0.27$	0.169	1.000	17/9/4
MCP $-$ VSA	$+0.00$	1.000	1.000	12/16/2
MCP $-$ Independent Controller	$+0.01$	1.000	1.000	12/15/3
Reflection $-$ VSA	-0.27	0.285	1.000	11/18/1
Reflection $-$ Independent Controller	-0.26	0.252	1.000	11/16/3
VSA $-$ Independent Controller	$+0.01$	1.000	1.000	17/9/4

Comparison with the baselines under ablation. On Sonnet 4.5, all four ablated configurations score below both MLR-Agent and AI Scientist V2 on Overall (Tables 1 and 3). The same holds on Opus 4.5, although the w/o Reflection configuration is only marginally below MLR-Agent. On Sonnet 4.6, the ablated configurations all remain above AI Scientist V2; only w/o VSA falls below MLR-Agent, while w/o MCP and w/o Reflection tie it and w/o Independent Controller stays above. Either core-state removal (w/o VSA or w/o Independent Controller) drops YouRA below both baselines by clear margins on two of the three backbones (Sonnet 4.5 and Opus 4.5).

Backbone dependence. Across all four ablations, the drops are largest on Sonnet 4.5 and Opus 4.5 and smallest on Sonnet 4.6 (backbone-drop columns of Table 16), so a stronger execution backbone partially compensates for a missing component. The compensation is incomplete. The full configuration is still the highest-scoring YouRA variant on every backbone, and even the smallest drop (w/o Independent Controller on Sonnet 4.6) leaves the ablated configuration below the full one.

Failure profiles under core-component ablation. Judge justifications for the two Sonnet 4.5 core-component ablations show distinct failure signatures aligned with each component's role. Under w/o VSA, recurring flags include numbers that disagree across sections and tables, code implementing different experiments than the paper describes, claims contradicting execution logs, and selective reporting of favorable runs, all consistent with losing the single persistent record connecting hypotheses, gates, and evidence links. Under w/o Independent Controller, judges repeatedly flag failed or crashed runs whose outputs are still reported as completed findings, validation gates relaxed post hoc instead of escalated, and pipelines that shrink far below their plans without re-attempt, consistent with losing independent stop/continue judgment and recovery escalation. In both configurations Soundness is the most damaged dimension while Clarity stays high, matching the fluent-but-unsupported failure mode that motivates YouRA's design (§1).

Note on Clarity Δ under ablation. On Sonnet 4.5, the no-MCP configuration leaves Clarity nearly unchanged (-0.05); the no-reflection configuration shows a small Clarity increase ($+0.075$). On Sonnet 4.6, both the MCP and reflection ablations slightly reduce Clarity and increase Novelty. The Sonnet 4.5 VSA and controller ablations follow the same pattern: Clarity decreases only slightly (-0.325 and -0.275) while Soundness and Significance drop sharply. Across all configurations, the Overall drops are driven primarily by Soundness and Significance, the dimensions in which VSA-conditioned and reflection-informed reasoning is most heavily exercised; surface-level clarity contributes little.