XYZ-Aquila
Agentic Intelligence System

AI4AI at Scale: A Full-Pipeline System for Enhancing LLM Agentic Capabilities

XYZ Agentic Team

July 22, 2026

Abstract

We present AI4AI, a bounded-exploration, verification-gated framework for improving complete agentic systems, and instantiate it in XYZ-Aquila, a family of two open-weight Deep Search agents built on Qwen3.6-35B-A3B and Qwen3.5-397B-A17B. Humans define a versioned optimization contract—the target capability, private development benchmark, admissible interventions, resource and risk bounds, and acceptance policy—while AI agents diagnose failures and develop scoped changes across data, learning, runtime, context management, tools, and infrastructure. An isolated evaluator supplies evidence to the acceptance gate without exposing hidden answers or labels; external benchmarks are withheld from routine optimization; and both accepted and rejected outcomes are retained with provenance in shared experience memory. The released systems combine graph-grounded, tool-reachable task construction, state-faithful supervised fine-tuning, and a fixed three-tool execution contract with explicit context policies and exact replay state. In the reported comparisons, XYZ-Aquila-mini attains the highest score in every column of the seven-benchmark sub-40B open-weight table, while XYZ-Aquila-pro leads every column of the six-benchmark sub-400B open-weight table and records the highest reported score among all listed systems on BrowseComp-ZH, LiveBrowseComp, and WideSearch. We release model weights, evaluation code, training details, representative traces, and versioned intervention records. Together, this case study provides evidence that bounded system-level optimization can produce strong matched-scale performance and a competitive parameter–performance frontier while keeping the improvement process inspectable, reviewable, and reproducible.

Figure 1: Visual overview across six of the seven agentic search benchmarks.

1 Introduction

Many complex technical tasks require coordinated improvement across models, tools, data, evaluation, infrastructure, and human expertise. Improving the agentic capabilities of large language models is one representative case: a capable agent must plan over long horizons, call tools, inspect evidence, recover from failures, obey task constraints, and interact with evaluators and infrastructure [1, 2, 3, 4]. In practice, these behaviors emerge from a coupled system in which base-model selection, data construction, supervised and reinforcement post-training, tool interfaces, verifier design, benchmark protocols, memory, logging, and human feedback all affect the final capability. Optimizing one component without understanding the others often produces brittle gains, hidden regressions, or improvements that do not transfer beyond the development setting.

AI-for-AI (AI4AI) offers a promising way to manage this complexity. Recent systems show that AI agents can assist with research, software engineering, experiment design, machine-learning workflows, fine-tuning, evaluation, and iterative debugging [5, 6, 7, 8, 9]. These successes suggest that AI agents can help improve other AI systems by diagnosing failures, proposing interventions, generating data, writing code, running experiments, and summarizing evidence. However, letting current LLM agents perform end-to-end agent improvement autonomously is still beyond their reliable capability [7, 6, 10]. The search space is large, the interactions among components are subtle, and naive automation can obscure why a change worked, overfit to available feedback, or blur responsibility for unsafe or invalid decisions [11]. End-to-end AI4AI therefore needs not only automation, but also transparency, accountability, and tractable human supervision.

We address this gap with a bounded-exploration AI4AI mechanism for complex agent development. Humans define the objective, private proxy benchmarks, resource and tool envelope, risk boundaries, and acceptance criteria. Multiple AI agents then operate inside this envelope: they diagnose failures, propose interventions, construct data, adjust post-training recipes, repair harnesses, analyze logs, and report evidence. An independent evaluator returns metrics and diagnostic signals without exposing answer keys or private labels, and external benchmarks remain reserved for final assessment. Candidate changes are adopted only when they clear the human-defined gate, while both accepted and rejected attempts are retained in an auditable record. The process is therefore recursive, but not open-ended: exploration is broad enough to improve the system, yet bounded enough that its decisions, evidence, and responsibilities remain inspectable.

A central part of the mechanism is human–AI coordination. Rather than asking humans to hand-approve every low-level step, we let human experts supervise the process through objectives, constraints, priorities, exception handling, and review of high-risk changes. AI agents keep the workflow tractable by routing tasks, preserving private-benchmark logs and reviewer comments as memory, summarizing experiment outcomes, surfacing relevant prior incidents, and asking for guidance when automated diagnosis is insufficient. This turns human judgment into a continuous supervisory signal while allowing multiple agents to carry out the operational work required by a large agent-development project.

As a case study, we apply the mechanism to Deep Search. Search agents are a high-signal testbed because they combine planning, tool use, evidence acquisition, source verification, failure recovery, and constrained synthesis in a measurable loop. The resulting systems are XYZ-Aquila-mini, a 35B-class agent based on Qwen3.6-35B-A3B, and XYZ-Aquila-pro, a 397B-class agent based on Qwen3.5-397B-A17B. The AI4AI loop selected each backbone and configuration using private-benchmark evidence. Figure 1 summarizes results on BrowseComp [12], BrowseComp-ZH [13], DeepSearchQA [14], LiveBrowseComp [15], Humanity's Last Exam [16], and WideSearch [17]; the full evaluation also includes GAIA [4]. XYZ-Aquila-mini leads the

35B-class comparison across the reported benchmarks, while XYZ-Aquila-pro is competitive with much larger systems, leading the large-model table on BrowseComp-ZH, LiveBrowseComp, and WideSearch and remaining close to the strongest comparators elsewhere. Because baseline scores come from heterogeneous public reports, harnesses, tool stacks, judges, and dates, these are benchmark-level comparisons rather than a single controlled total order.

The report makes three contributions.

1. We formulate bounded-exploration AI4AI as a practical mechanism for complex agent development: humans set the envelope, multiple AI agents search within it, independent evaluation gates adoption, and every cycle writes accepted and rejected evidence into durable memory.

2. We show how human experts can coordinate with AI agents through objectives, constraints, shared memory, progress reports, and exception-based review, making the improvement process tractable, transparent, and accountable without requiring manual approval of every step.

3. We present XYZ-Aquila-mini and XYZ-Aquila-pro as a Deep Search case study. Their benchmark results, together with accepted and rejected interventions across data, post-training, reinforcement learning, harness, and infrastructure surfaces, demonstrate that bounded AI4AI can produce strong agentic performance while preserving auditability.

The remainder of this report is organized as follows. Section 2 formalizes the bounded AI4AI process, including the screened RL extension, human–AI orchestration, and optimization surfaces. Sections 3 and 4 present the search-agent instantiation and empirical results. Section 5 discusses auditability, broader applicability, limitations, and future work, before Section 6 concludes.

1.1 Related Work

Tool-Using and Web-Browsing Language Agents Early web-augmented language-agent work showed that models can be trained to interact with browser-like environments rather than answer only from parametric knowledge. WebGPT combined demonstrations, comparison data, and reward modeling for long-form web question answering [1]; ReAct interleaved reasoning traces with actions for knowledge-intensive and interactive tasks [2]; and Toolformer showed that models can learn when and how to call external tools from self-supervised API-use annotations [3]. These works motivate treating browsing, tool use, and reasoning as coupled behaviors, which AI4AI then optimizes as part of a larger agentic system.

Agentic Post-Training for Long-Horizon Systems Recent agent-focused systems increasingly use post-training to teach long-horizon interaction, tool use, evidence verification, and recovery from failed trajectories. MiroThinker-1.7 and MiroThinker-H1 emphasize structured planning, tool interaction, and local/global verification [18]. Agents-A1 studies agent-horizon scaling for a 35B MoE agent with long-horizon knowledge-action infrastructure and multi-stage training [19]. Nex-N2 reports open-source Pro and mini variants for search, coding, tool calling, terminal execution, and reasoning [20], while Apodex-1.0 emphasizes deep-research post-training, evidence-audited operation, and public harness documentation [21, 22]. Our contribution is complementary: we focus on the bounded improvement loop that selects, audits, and records interventions across the whole system.

AI4AI for LLM Training and System Optimization AI4AI uses AI to assist the design, training, evaluation, and deployment of AI systems themselves. Autonomous research systems such as the AI Scientist and Agent Laboratory connect idea generation, implementation, experimentation, and reporting [5, 6]; MLE-bench and AIDE evaluate machine-learning-engineering tasks and code-solution search [7, 8]; and systems such as ML-Master, R&D-Agent, Reasoning as Gradient, FT-Dojo, and Agent^2 RL-Bench study iterative exploration, memory, fine-tuning, or RL-loop closure under fixed budgets [9, 23, 24, 25, 10]. These systems show the promise of AI-assisted optimization; our emphasis is on making that recursion bounded by hidden evaluators, scoped interventions, retained negative evidence, and human-defined acceptance criteria.

This view subsumes rather than replaces established preference optimization. InstructGPT combines supervised fine-tuning, reward modeling, and RL from human preferences; Constitutional AI supplements human supervision with AI feedback; and DPO directly optimizes a policy from preference data [26, 27, 28]. For agentic systems, however, preference optimization is only one surface. A bounded AI4AI loop must also diagnose traces and private-benchmark logs, audit data and tool-use interventions, repair evaluation harnesses, manage experiment memory, and escalate uncertain choices to humans. Reflexion and ML-Master motivate linguistic feedback and selectively retained experience for this purpose [29, 9]; XYZ-Aquila applies the idea to the full Deep Search stack.

2 AI4AI: Bounded System Optimization for Agentic Intelligence

We introduce *AI-for-AI (AI4AI)*, a bounded, verification-gated framework for improving agentic intelligence through system-level optimization. Its starting point is that *agentic intelligence* is a property of a complete agentic system rather than of an isolated model checkpoint: it emerges from coupled choices over models, learning procedures, policies, tools, data, evaluation, and infrastructure. Within AI4AI, an AI optimizer diagnoses the system and proposes scoped interventions; humans define the objective, admissible search envelope, risk boundaries, and acceptance policy; and an isolated evaluator supplies the evidence used to gate adoption. The resulting process is recursive but controlled: each cycle can improve the incumbent system, while both accepted and rejected outcomes are retained as auditable evidence for later cycles. In this section, we first formalize the agentic-system design space, then define the bounded AI4AI optimization loop, and finally describe how human expertise and authority enter the process.

2.1 Agentic Intelligence as a System-Design Problem

Let A_θ denote an agentic system instantiated by a coherent configuration θ. The configuration specifies not only a model checkpoint, but also the learning procedures, control policies, tools, data pipelines, development diagnostics, and infrastructure that jointly determine the system's behavior. We write the admissible design space as

$$\theta \in \Theta, \qquad \Theta \subseteq \Theta_{\text{model}} \times \Theta_{\text{learn}} \times \Theta_{\text{policy}} \times \Theta_{\text{tool}} \times \Theta_{\text{data}} \times \Theta_{\text{dev-eval}} \times \Theta_{\text{infra}}. \tag{1}$$

The product notation groups the main design families, while the subset relation captures compatibility, implementation, resource, and safety constraints. Thus, θ is one coherent system design rather than an arbitrary Cartesian combination of independent choices. The design space includes, but is not limited to:

1. model components and configurations (Θ_{model}), including the base model, adapters, context window, decoding behavior, and system prompts;
2. post-training and learning procedures (Θ_{learn}), including supervised fine-tuning, preference optimization, reinforcement learning, distillation, and related adaptation strategies;
3. retrieval, reasoning, and control policies (Θ_{policy}), including planning strategies, memory access, retrieval rules, stopping criteria, and tool-selection policies;

4. tool definitions and schemas (Θ_{tool}), including available APIs, argument schemas, permissions, and tool documentation;
5. data generation and processing pipelines (Θ_{data}), including synthetic-data construction, filtering, retrieval corpora, and preprocessing rules;
6. development-time evaluation harnesses ($\Theta_{\text{dev-eval}}$), including optimizer-visible diagnostic benchmarks, judges, logging, tracing, and aggregation procedures, but excluding the isolated gate evaluator E_{gate}, which remains outside θ; and
7. system infrastructure (Θ_{infra}), including executors, sandboxes, serving stacks, caches, compute budgets, latency controls, and reliability mechanisms.

Agentic search, our running example, makes this system-level view concrete. A search agent must decompose a query, plan and revise a retrieval strategy, interact with external tools and information sources, preserve relevant evidence, recover from failed branches, and produce a constrained final answer over dozens or even hundreds of steps. An early decision about planning, retrieval, tool use, or context management can alter the entire downstream trajectory. Capability therefore arises from interactions among components, not from any component in isolation.

Two consequences follow. First, the design space is combinatorially large: each family is itself high-dimensional, and a complete configuration must make coupled choices across all of them. Second, the interactions are non-separable: a data intervention can change the effectiveness of a training recipe, which can change the policy's behavior under a particular harness and infrastructure stack. Manual tuning and static recipes therefore cover only a small fraction of the admissible space. These properties motivate an AI-assisted search process, provided that exploration remains contract-bounded, independently evaluated, and auditable.

2.2 Bounded-Exploration AI4AI

We formulate AI4AI as a governed sequential search process over system interventions. Each cycle starts from an accepted incumbent configuration, proposes a bounded change, produces a candidate system, evaluates that candidate through an isolated gate, and records both the evidence and the resulting decision. Figure 2 summarizes the optimization contract, the four-stage cycle, and the shared experience memory $\mathcal{F}_t$ that connects successive cycles.

[!t]

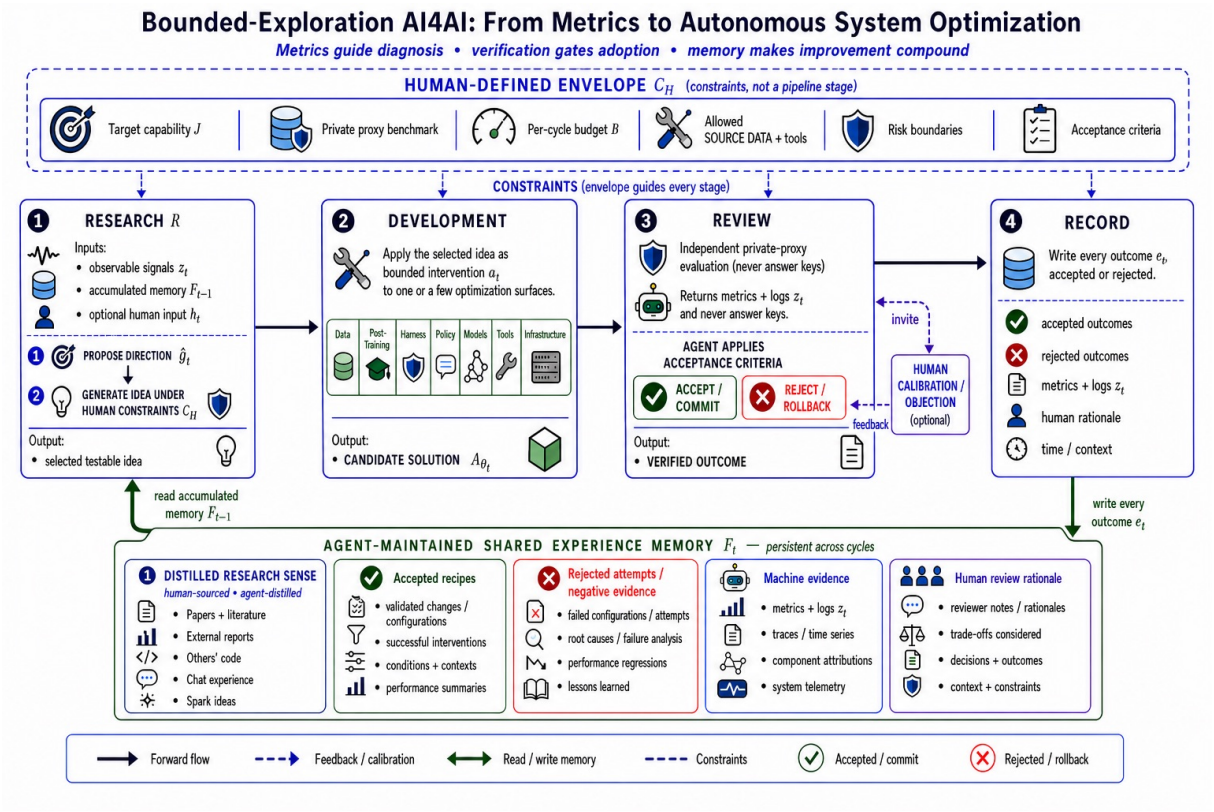

Figure 2: The bounded-exploration AI4AI loop. A human-defined optimization contract governs a repeated cycle of *Research*, *Development*, *Review*, and *Record*; accepted and rejected outcomes are retained in the shared experience memory $\mathcal{F}_t$ as evidence for later cycles. The Research stage is expanded in Figure 3.

The top panel of Figure 2 specifies the human-defined contract: the target capability, private proxy benchmark, resource envelope, allowed data and tools, risk boundaries, and acceptance criteria. The central panel shows the Research–Development–Review–Record cycle. The shared experience memory $\mathcal{F}_t$ retains machine-generated evidence, human expertise, accepted recipes, and rejected attempts, while the bottom panel expands the Research stage into diagnosis, proposal generation, and intervention selection.

2.2.1 Optimization Contract and Separation of Roles

Human-defined optimization contract. At cycle t, optimization operates under a human-defined versioned contract

$$\mathcal{H}^{(\nu_t)} = \big(J, \mathcal{B}_{\text{priv}}, E_{\text{gate}}, \mathcal{C}_H, G, B_{\text{cycle}}, \tau_{\text{stop}}\big), \tag{2}$$

where ν_t identifies the contract version active during cycle t; any authorized change to a contract component creates a new version. Here, J is the intended capability to improve; $\mathcal{B}_{\text{priv}}$ is the private proxy benchmark used for development; E_{gate} is the isolated evaluator used at cycle boundaries; $\mathcal{C}_H$ is an admissibility predicate over interventions and configurations; G is a versioned acceptance policy; B_{cycle} is the per-cycle resource and evaluation budget; and τ_{stop} specifies when the process must stop, pause, or request reauthorization. Collectively, these terms define what may be changed, what evidence may be used, what risks are disallowed, and what evidence is sufficient for adoption.

Separation of roles. Let O_ϕ denote the AI4AI optimizer and A_θ the target system. Their responsibilities are deliberately separated. The optimizer diagnoses failures, proposes interventions, implements candidates, and summarizes evidence. The evaluator E_{gate} measures candidate behavior without exposing private answers or labels. The acceptance policy G maps that evidence to an adoption decision. Humans define and version the contract, authorize changes to it, and resolve cases that are escalated because they are high-risk, ambiguous, or outside the current envelope. The optimizer cannot modify E_{gate} or G within the same decision path in which they are used to evaluate its proposal. A change to the evaluator or gate may itself be proposed, but activating such a change requires a separate validation path, explicit authorization, and re-baselining under the new version.

Objective, evidence, and decision. The conceptual target is an admissible system with high general capability:

$$\Theta_H = \{\theta \in \Theta : \mathcal{C}_H(\theta) = 1\}, \qquad \theta^\star \in \arg\max_{\theta \in \Theta_H} J(A_\theta). \tag{3}$$

This equation states the intended optimization problem; it is not an operational claim that J is directly observable or globally optimized. The method distinguishes three objects: $J(A_\theta)$, the intended capability; z_t, the observable evidence returned by the private evaluation process; and d_t, the adoption decision produced by the gate. Private-benchmark gains therefore constitute development evidence, not proof of transfer. Transfer is assessed only through external evaluation that remains outside the optimization state and is queried solely for final or explicitly authorized assessment.

2.2.2 The Research–Development–Review–Record Cycle

At the start of cycle t, θ_t denotes the accepted incumbent configuration, $\mathcal{F}_t$ the durable and versioned shared experience memory, and h_t any optional human input relevant to the current cycle. The memory contains complete cycle records together with contextualized experience retained for later retrieval. The four stages in Figure 2 define one state transition.

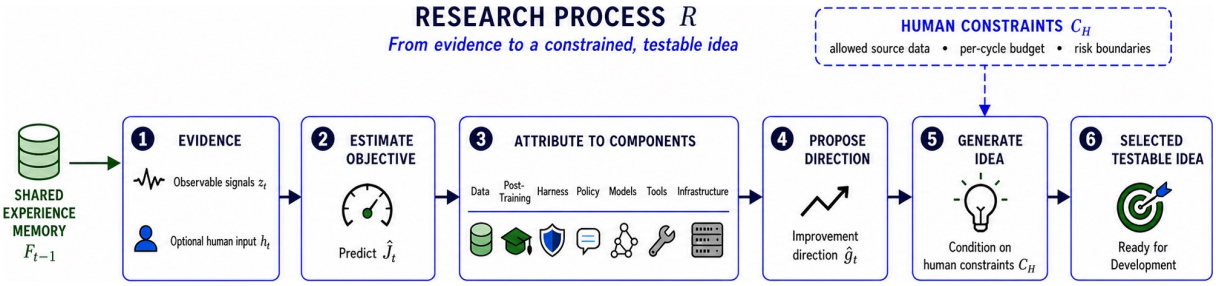

Figure 3: The Research stage of the AI4AI cycle. From the incumbent configuration θ_t, the shared experience memory $\mathcal{F}_t$, and any human input h_t, the Research module $\mathcal{R}_\phi$ diagnoses the current system, generates candidate proposals, and selects an admissible intervention a_t to hand to Development.

Research. The Research module $\mathcal{R}_\phi$ within O_ϕ diagnoses the incumbent and proposes an admissible intervention:

$$(a_t, \widehat{u}_t, r_t) = \mathcal{R}_\phi(\theta_t, \mathcal{F}_t, h_t; \mathcal{H}^{(\nu_t)}), \qquad a_t \in \mathcal{A}_H(\theta_t), \qquad c(a_t) \le B_{\text{cycle}}. \tag{4}$$

Here $\mathcal{A}_H(\theta_t)$ is the set of interventions permitted by the active contract, $\widehat{u}_t$ is the predicted utility of the proposal, and r_t records its diagnosis and rationale. Research may use optimizer-visible trajectories, development diagnostics, prior private-benchmark summaries, reviewer notes, and retrieved experience from $\mathcal{F}_t$, but it cannot access hidden labels or the external evaluation suite.

Development. The Development stage implements the intervention without querying the gate evaluator:

$$\widetilde{\theta}_t = f(\theta_t, a_t), \qquad \mathcal{C}_H(\widetilde{\theta}_t) = 1. \tag{5}$$

The resulting $\widetilde{\theta}_t$ is a candidate rather than an accepted system state. An intervention may modify one design family or a small, explicitly scoped set of coupled families, but it must remain within the permissions, resource envelope, and risk boundaries of $\mathcal{H}^{(\nu_t)}$.

Review. At the cycle boundary, the isolated evaluator produces a versioned evidence bundle

$$z_t = E_{\text{gate}}(A_{\widetilde{\theta}_t}; \mathcal{B}_{\text{priv}}), \tag{6}$$

which may contain aggregate scores, difficulty splits, cost and latency measurements, candidate-generated trajectories, and approved failure diagnostics, but never the private answer key. Let z_t^{ref} denote the corresponding evidence for the current approved reference under the same evaluator version. The versioned gate then returns

$$d_t = G(z_t, z_t^{\text{ref}}, a_t; \mathcal{H}^{(\nu_t)}) \in \{\text{accept}, \text{reject}, \text{escalate}\}. \tag{7}$$

For resolved accept/reject cases, the incumbent is updated by

$$\theta_{t+1} = \begin{cases} \widetilde{\theta}_t, & d_t = \text{accept}, \\ \theta_t, & d_t = \text{reject}. \end{cases} \tag{8}$$

An escalated cycle does not advance until an authorized human resolves it. The resolution, including any override or contract amendment, is recorded as part of the cycle evidence. Any amendment creates a new contract version, and judging the candidate under that version requires a separate evaluation rather than changing the active contract within the current decision. Thus the optimizer proposes and implements, the evaluator measures, the gate decides, and humans govern the contract and exceptions.

Record. Every cycle produces an evidence record, regardless of whether its candidate is adopted:

$$e_t = (\theta_t, a_t, \widetilde{\theta}_t, z_t, d_t, r_t, \nu_t, v_t), \tag{9}$$

where v_t contains provenance such as code, data, prompt, model, harness, evaluator, and normalization versions; resource use; timestamps; representative traces; and reviewer notes. The outcome is then committed to the shared experience memory:

$$\mathcal{F}_{t+1} = \text{Update}(\mathcal{F}_t, e_t). \tag{10}$$

The update is durable and versioned: it preserves the complete cycle record and, when the evidence is sufficiently supported, creates or revises a contextualized entry for later retrieval. Accepted candidates may become reusable recipes; rejected candidates may become negative evidence, but only together with the conditions under which they failed. Thus $\mathcal{F}_t$ is one shared memory that supports both auditability and compounding experience, without treating every raw record as an equally reliable lesson.

External evaluation. External benchmarks are not part of O_ϕ's observation history, $\mathcal{F}_t$, or the gate feedback used for routine optimization. They are reserved for final or explicitly authorized assessment. Their role is therefore to test whether the development evidence transfers beyond the private loop, not to provide another adaptive search signal.

2.2.3 Boundedness, Evidence Discipline, and Compounding Shared Memory

What is bounded. Boundedness refers to constrained authority, evidence access, and resource use rather than to a small design space. The contract imposes five complementary boundaries:

1. **Scope bounds:** only named design surfaces and intervention classes may be changed;
2. **Permission bounds:** only authorized data sources, tools, models, and infrastructure may be used;
3. **Evaluation bounds:** hidden labels, answer keys, and external benchmarks remain inaccessible to the optimizer, and the active gate cannot be changed within the decision it governs;
4. **Resource bounds:** each cycle is limited by compute, wall-clock time, rollout, and evaluation budgets; and
5. **Authority bounds:** high-risk, ambiguous, or out-of-envelope cases must be rejected or escalated rather than resolved through optimizer discretion.

The framework does not require a fixed number of cycles, but it is not open-ended. Continued operation is permitted only while the active contract remains valid. The stopping rule τ_{stop} pauses or terminates the process when contract-specified conditions are met, such as budget exhaustion, a deadline, a performance plateau, repeated gate rejection, elevated risk, or explicit human termination; further exploration then requires reauthorization.

Evidence discipline and low-cost screening. Keeping private labels hidden reduces direct leakage, but it does not by itself guarantee transfer: repeated adaptation to a fixed development suite can still produce adaptive overfitting. We therefore isolate the evaluator, limit each candidate to one gate query at the cycle boundary, retain benchmark and evaluator versions, inspect near-duplicate and leakage risks, and reserve the external suite for non-adaptive assessment. For expensive intervention classes, Development may first produce a low-cost pilot artifact p_t before committing the full training or systems budget:

$$s_t = E_{\text{screen}}(p_t), \qquad \text{pass}(s_t) = 1 \Longrightarrow a_t \text{ is eligible for full development and } E_{\text{gate}}. \tag{11}$$

Screening may prune an expensive proposal, but it cannot authorize adoption. Full acceptance still requires the independent evidence and policy in Eqs. (6)–(7). In the Deep Search instantiation, reinforcement-learning interventions are one example of proposals that can require such screening before a costly training-and-evaluation run.

Why the loop can compound. The loop compounds only when past outcomes are retained with enough context to determine when they remain applicable. The update rule in Eq. (10) therefore attaches provenance, evaluator and system versions, applicability conditions, confidence, and validation status to retained experience in $\mathcal{F}_t$. Accepted interventions are not treated as universal recipes, and rejected interventions are not treated as universally invalid; conflicting or stale lessons remain traceable in the shared memory and can be revised or retired from active retrieval while their underlying records remain preserved. With these safeguards, later Research stages can retrieve relevant evidence, avoid repeating known failures, and compare new proposals against prior interventions, reducing redundant trials without assuming guaranteed convergence.

The contract and gate determine what may be adopted. The remaining question is how human expertise, priorities, and exceptions enter the optimizer's information state without requiring manual approval of every low-level action.

2.3 Human–AI Collaboration Paradigms

Human–AI collaboration serves two distinct functions in the bounded loop. First, humans exercise authority over the optimization contract: they set objectives and constraints, authorize

contract changes, and resolve escalated cases. Second, they contribute domain knowledge that improves diagnosis and proposal selection inside the active contract. The optimizer carries out the operational work of analysis, implementation, experimentation, and reporting, while the gate continues to govern adoption.

A shared collaboration interface. We implement these information flows through a shared chat room. A persistent orchestration agent is present in a channel shared by the relevant researchers; teammates can @-mention it to delegate a task, contribute evidence, question a result, or redirect an active thread. The orchestration agent can execute bounded tasks directly or route them to specialized agents, while reporting progress in the originating thread. Interaction roles in the room are flexible, but authority boundaries are not: participants may propose and discuss freely, whereas only authorized humans may change $\mathcal{H}$, and only the gate may authorize routine adoption. Every message is incorporated into the durable, versioned shared experience memory $\mathcal{F}_t$, preserving the exchange for audit; distilled, contextualized, and validated items are additionally marked as reusable experience for later retrieval.

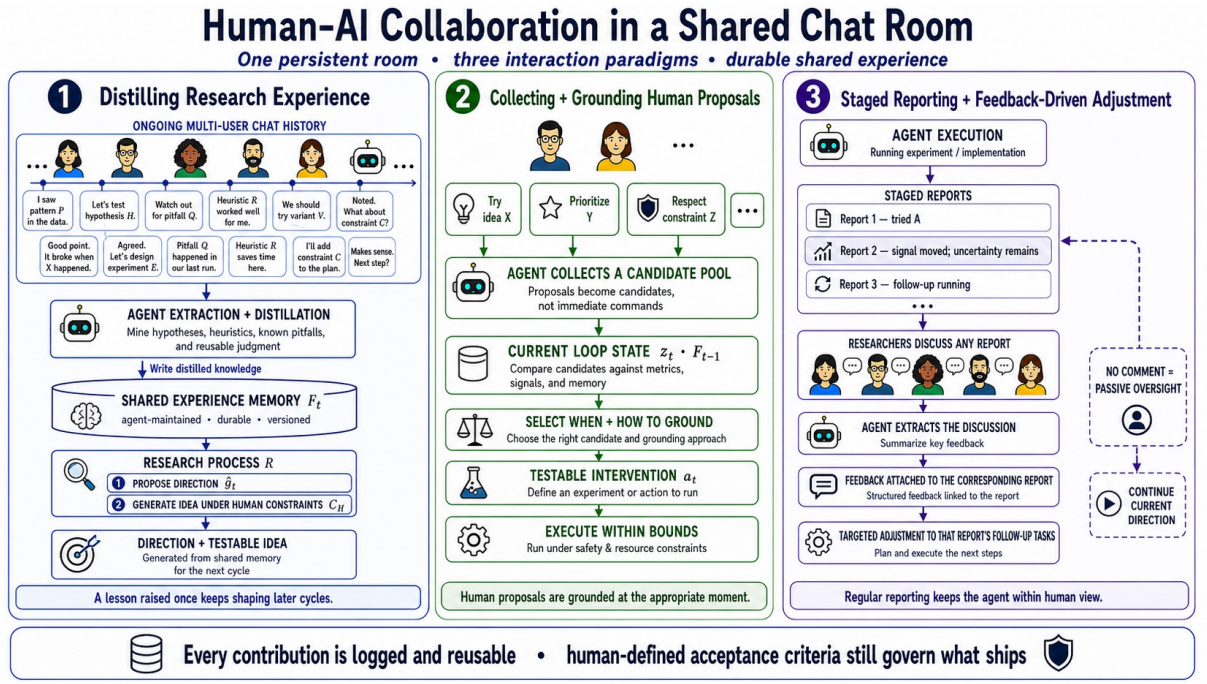

Figure 4: The three human–AI collaboration paradigms in the shared chat room: (1) distilling research experience from ongoing discussion into shared memory, (2) collecting human proposals as candidates and grounding them into interventions, and (3) staged progress reporting under human review with feedback bound back to each report.

Paradigm 1: Distilling research experience from discussion. Much of a team's research experience lives in ordinary conversation—related work, findings from human-run experiments, and debates over what to try. The agent continuously mines this discussion, distilling the reusable judgment it contains (hypotheses, heuristics, known pitfalls) into the shared memory $\mathcal{F}_t$ and consulting it when later choosing the improvement direction $\hat{g}_t$ (Figure 4). This targets the agent's largest present gap—the seasoned judgment of an experienced researcher—without asking anyone to file knowledge in a special form.

Paradigm 2: Collecting and grounding human proposals. Researchers also post concrete proposals—an idea to try, a direction to prioritize, a constraint to respect. The agent

treats these as *candidates* rather than immediate orders (Figure 4): it collects them, weighs them against the loop's current state, and grounds the right ones into testable interventions a_t at the right moment, honoring human input without abandoning what it is already pursuing.

Paradigm 3: Staged reporting and feedback-driven adjustment. The orchestration agent posts staged reports that state what was attempted, what evidence changed, what remains uncertain, and what decision is pending. Human discussion is attached to the relevant report and becomes cycle-local feedback, allowing a researcher to correct a diagnosis or redirect downstream work without rewriting the entire memory state. The absence of a reply does not constitute approval; it simply leaves the active contract, gate, and current task plan unchanged. Explicit review is required whenever the contract or escalation policy requires it.

Exception-based control and traceability. Ordinary in-scope candidates can be accepted or rejected by the versioned gate without per-cycle human sign-off. High-risk, ambiguous, or out-of-envelope candidates must be rejected or escalated, and any change to the objective, evaluator, gate, permissions, or risk boundary requires explicit human authorization. Each adopted change is therefore traceable to a scoped proposal, an implemented candidate, versioned evidence, a gate decision, and any human intervention that affected the outcome. Human–AI interaction is not merely a convenience layer: it is the mechanism through which scarce research judgment enters the loop while governance remains explicit.

The output of bounded AI4AI is consequently not only an improved incumbent A_{θ_T}, but also a durable shared experience memory $\mathcal{F}_T$ containing auditable cycle records and reusable, contextualized lessons. Section 3 instantiates this framework for Deep Search across data construction, supervised fine-tuning, reinforcement learning, and harness design.

3 Agentic Search as a Case Study

Section 2 formulated agentic intelligence as a system-design problem and AI4AI as an outer optimization loop over that system. Agentic search provides a concrete instantiation because a single answer may require long-horizon planning, repeated tool use, evidence acquisition and verification, recovery from false leads, and constrained synthesis. It also produces long trajectories whose actions, observations, context transformations, and final decisions can be replayed and audited.

XYZ-Aquila is therefore a search-agent configuration rather than an isolated checkpoint. Each released system combines a selected backbone with a data-construction pipeline, a supervised post-training recipe, an execution runtime, a context-management policy, replay and logging infrastructure, and a separately versioned evaluation gate. Within each model scale, backbone selection was itself an empirical AI4AI decision under the private development gate, yielding Qwen3.6-35B-A3B for XYZ-Aquila-mini and Qwen3.5-397B-A17B for XYZ-Aquila-pro.

Two logical levels should be distinguished. The *outer* AI4AI loop diagnoses the incumbent, proposes a scoped intervention, develops a candidate, and accepts or rejects it using the contract in Section 2. The *inner* search-agent learning pipeline contains SFT and, when separately authorized, RL. AI4AI therefore does not prescribe a mandatory SFT-to-RL sequence: data, learning, runtime, evaluation, or infrastructure changes may be proposed in different cycles and are gated as separate interventions. The remainder of this section first instantiates the Deep Search contract, then presents four representative intervention families, and finally states which components define the released systems.

3.1 Deep Search Instantiation and Optimization Contract

At inference time, the agent can call three executable tools: `search`, `scrape`, and `python`. Final-answer submission is handled by the interaction protocol rather than by an additional tool. The private development suite supplies versioned aggregate metrics, difficulty splits, cost and reliability measurements, and approved trace diagnostics, while answers, evaluator-side evidence, normalization rules, and private labels remain hidden from the optimizer and rollout agent.

Table 1: Instantiation of the bounded AI4AI contract for Deep Search.

Contract element	Deep Search instantiation
Target capability	Long-horizon, evidence-grounded search under fixed interaction and resource bounds.
Private development evidence	A hidden internal QA suite, versioned trajectories, failure splits, and cost/reliability diagnostics.
Admissible interventions	Backbone selection within a scale; task and trajectory data; supervised learning; runtime and context policy; replay and infrastructure; separately screened RL proposals.
Fixed agent-facing interface	`search`, `scrape`, and `python`, with fixed tool semantics, final-answer protocol, visible-context limit, and turn budget.
Evaluation separation	Evaluator-side answers and labels are hidden; the active gate evaluator and acceptance policy cannot be modified within the decision path they govern.
External assessment	Public benchmarks remain outside the optimization state and are used only for final or explicitly authorized evaluation.

Three invariants connect all interventions below. First, the *answer-hidden contract* exposes only the released question and fixed tools during rollout. Second, candidate changes must preserve the agent-facing interface and resource envelope, so an apparent gain cannot be attributed to additional tools, a larger visible context, or a larger turn budget. Third, every supervised or analyzed decision is tied to the exact state visible to the agent at that step, denoted by V_l, rather than to the fuller append-only audit history available to developers. These invariants make data, learning, and runtime changes comparable under one bounded contract.

3.2 Representative AI4AI Interventions Across the Search Stack

The cases below are organized by intervention family rather than by a mandatory chronology. A data intervention may reveal a runtime defect; a harness change may invalidate previously collected trajectories; and an expensive learning proposal may be screened without being adopted. Each case follows the same logic: diagnosis, intervention, bounds and review, and the lesson retained in the shared experience memory.

3.2.1 Case A: Graph-Grounded Task Construction

Diagnosis. Ordinary question–answer pairs do not reliably exercise the behaviors required by Deep Search. Data audits and replay exposed several confounds: some questions collapsed into a single keyword lookup; some had ambiguous aliases or non-unique answers; some were supported by an offline evidence graph whose decisive pages were not recoverable through the actual tools; and some appeared multi-hop while only one clue was necessary. Training on such records would reward answer recall or dataset artifacts rather than search, disambiguation, and constraint checking.

Intervention. We construct candidate tasks from connected evidence graphs built over approved sources. Nodes represent pages or evidence units, and edges represent auditable relations such as hyperlinks, citations, shared entities, or temporal connections. A generator samples a connected subgraph and produces a seed question, reference answer, supporting path, answer constraints, source lineage, expected difficulty, and normalization rule. Direct anchors—for example entity names, titles, dates, or locations—are then replaced with relational descriptions or indirect clues so that solving the task requires evidence localization and joint reasoning rather than a single lookup. The graph and all evaluator-side metadata are used only for construction, validation, and audit; they are never exposed to the rollout agent.

Bounds and review. A candidate record is retained only if its evidence is reachable with the fixed `search` and `scrape` tools, its answer is supported and unique under the stated normalization rule, and query-only shortcut tests do not recover the answer without the intended evidence path. Reviewer audits additionally check ambiguous aliases, false-lead wording, and graph links that are valid offline but brittle in the runtime. Rejected categories are written back to the shared memory and used to revise source sampling, generation prompts, obfuscation prompts, validation thresholds, and reviewer checklists. This graph-grounded construction and validation recipe is an adopted component of the released data pipeline.

Retained lesson. Task difficulty must be defined relative to the deployed tool contract: an offline-valid question is not a valid search task unless its evidence is reachable, its answer is identifiable, and its shortcut surface is controlled under the runtime the agent will actually use.

3.2.2 Case B: State-Faithful Supervised Fine-Tuning

Diagnosis. A correct final answer is not sufficient evidence that an entire long trajectory should be imitated. A successful rollout may contain malformed assistant turns, schema-invalid tool calls, degenerate repetitions, or local protocol failures. Conversely, a valid tool call may return a timeout or an unavailable page, and the subsequent recovery behavior is useful supervision. Most importantly, the append-only audit log can contain evidence that was folded, summarized, truncated, or otherwise absent from the model's rendered context. Training against that fuller history would create a train–inference mismatch.

Intervention. Accepted tasks from Case A are executed in the fixed ReAct-style runtime, and each rollout stores the complete action–observation history, every context-management operation, and the exact agent-visible state V_l at each assistant decision. The converter reconstructs V_l under the active policy—including full-context rendering, tool-response folding, and summary-based compaction—and uses that state as the conditioning context for the supervised target. Valid reasoning, tool calls, and final-answer turns contribute to the loss; malformed assistant turns and repeated invalid loops remain available as context when needed but are masked from supervision. Valid calls that encounter tool-side failures are retained so that the model can learn recovery. Training uses assistant-token-level normalization and long-sequence packing up to 256K tokens; scale-specific hyperparameters are reported in the reproducibility appendix.

Bounds and review. Rollouts receive only the released question and the fixed tool interface. Reference answers, supporting evidence, answer constraints, lineage, and normalization remain evaluator-side. A trajectory is eligible only when the final answer is valid, its support is present in evidence visible to the agent, and the trace exhibits the behaviors intended for imitation, including query decomposition, targeted retrieval, inspection of relevant pages, recovery from

false leads, and final constraint checking. The exact-state reconstruction, turn-level masks, and approved mixture recipe were retained in the released SFT pipeline.

Retained lesson. Supervision should target the action taken from the state the agent actually observed, not from the more complete state available to the auditor. This distinction turns replay fidelity into a training requirement rather than only a debugging convenience.

3.2.3 Case C: Runtime, Context Management, and Replay

Diagnosis. The same wrong final answer can arise from different system failures: the agent may issue a poor query, the search backend may return unstable results, a page may be extracted incorrectly, decisive evidence may be removed by context management, or visible evidence may be ignored during finalization. Without a reliable runtime and a separation between the full trace and the rendered state, these causes are confounded, making both diagnosis and training-data conversion unreliable.

Intervention. The execution harness preserves a fixed three-tool interface while improving reliability within that interface. `search` returns a fixed top-10 result set, placing the burden on query formulation and source selection rather than on a model-chosen retrieval depth. `scrape` routes source types through extraction paths designed to preserve decisive content, using document-preserving paths when feasible and fixed chunked extraction and merging for long pages. `python` runs in a stateful per-task sandbox so that imports, files, variables, and lightweight command execution persist across valid calls without exposing private benchmark data or an implicit judging channel.

Long trajectories are rendered through explicit context policies, including full-context mode, tool-response folding, summary-based compaction, scrape-result compression, finalization routing, and discard-all reset. These policies may change what is visible at a later step, but they do not add tools, enlarge the context window, or increase the turn budget. Every run therefore stores both an append-only audit log and the exact V_l rendered at each decision. If decisive evidence exists in the audit log but not in V_l, the failure is attributable to context rendering; if it is present in V_l but omitted from the answer, the failure is more likely finalization or constraint checking.

Bounds and review. Runtime candidates are compared under the same tool semantics, visible-context limit, turn budget, and final-answer protocol. A runtime change is admissible only if it improves execution or observability without strengthening the task interface. We also distinguish the execution-and-replay harness, which is part of the candidate system, from the active gate evaluator, which consumes versioned outputs but remains isolated under Section 2. Changes to judge prompts, answer normalization, or benchmark-specific scoring rules require a separate validation path and re-baselining; they cannot be activated inside the same decision in which they would favor a candidate. The retained runtime, context, and replay mechanisms are adopted components of the released configuration.

Retained lesson. For long-horizon agents, observability is part of capability engineering. Exact visible-state logging makes failures attributable and lets the same trajectory support evaluation, debugging, and state-faithful training without conflating those roles.

3.2.4 Case D: Screened Reinforcement-Learning Extension

Diagnosis. Outcome-only verifiable rewards are clean but sparse for long-horizon search. A terminal score does not identify whether query reformulation, evidence gathering, disambigua-

tion, or finalization was decisive, and full RL experiments require expensive rollouts, reward computation, and policy updates. Under the bounded contract, such proposals should first be screened rather than treated as a default second stage of every AI4AI cycle.

Intervention. The screened RL task pool is disjoint from the SFT records and is biased toward medium- and high-difficulty questions, preserving failure surface after supervised training. For a logged visible state V_l, we form an answer-conditioned teacher by revealing the reference answer only as evaluator-side auxiliary context and compute the attribution score

$$a_l = \mathrm{KL}(\pi_\theta^{\mathrm{ans}}(\cdot \mid V_l, r_i) \,\|\, \pi_\theta(\cdot \mid V_l)) \,.$$

The score measures how strongly access to the answer changes the policy at that state. Qualitative screening indicated that high-a_l turns concentrate around information gathering, disambiguation, and finalization, making the signal useful for process-level attribution rather than as a standalone success criterion. To reduce the systems cost of training on context-managed trajectories, exact shared prefixes can be merged with heterogeneous attention masks so that repeated prefix computation is reused while each sample preserves its rollout-time visible context [30, 31]. The motivation for denser credit assignment follows prior work on verifiable-reward and agentic RL optimization [32, 33].

Bounds and review. The answer remains hidden from the rollout policy, the agent-facing tools and budgets remain fixed, and the KL signal is evaluated only as a candidate process guide. In the study reported here, this design passed low-cost screening but did not proceed through a full training-and-gate evaluation sufficient for adoption. We therefore report it as a screened extension and do not count it among the components defining the released XYZ-Aquila-mini or XYZ-Aquila-pro systems.

Retained lesson. Low-cost screening can prioritize expensive learning ideas and preserve their evidence without confusing technical promise with adoption. In the bounded loop, passing a pilot is permission to run a full experiment, not permission to ship.

3.3 Resulting Configuration and Evaluation Handoff

The reported release separates adopted system components from proposals that were only screened. XYZ-Aquila-mini uses the Qwen3.6-35B-A3B backbone, while XYZ-Aquila-pro uses Qwen3.5-397B-A17B. Both systems use the graph-grounded and tool-reachable task-data pipeline, state-faithful SFT with turn-level masks and long-sequence packing, and the fixed three-tool execution contract with explicit context policies and exact replay state. Their learning rates, batch sizes, and other scale-specific training settings are reported in the reproducibility appendix. The answer-conditioned RL design in Case D is not part of either reported release because it did not complete the full training-and-gate path required for adoption.

The released artifact is thus the composition of a selected backbone and the adopted data, SFT, runtime, context, and replay interventions; it is not the union of every proposal explored by AI4AI. Accepted and rejected records, including the screened RL pilot, remain in the shared experience memory with their provenance and decision status. Because the external benchmark suite was excluded from that optimization state, the evaluation that follows tests whether the private development evidence transfers beyond the loop rather than supplying another adaptive search signal.

4 Evaluation

This section evaluates the two search agents produced by the bounded AI4AI process in Sections 2 and 3. The external benchmark suite reported here was excluded from the optimization state and was not used as routine feedback during development; all optimization decisions were made using the private development benchmark and its isolated gate. The purpose of this section is therefore to test whether the final systems transfer beyond the development loop.

Across the reported comparisons, **XYZ-Aquila-mini** obtains the highest score in every benchmark column of the small-scale open-weight table, while **XYZ-Aquila-pro** obtains the highest score in every column of the sub-400B open-weight table. Against the broader set of frontier-scale or proprietary systems listed here, XYZ-Aquila-pro also records the highest reported score on BrowseComp-ZH, LiveBrowseComp, and WideSearch, and remains near the strongest reported results on BrowseComp and DeepSearchQA. Because some baseline scores are reproduced in our harness whereas others are collected from heterogeneous public reports, these results should be interpreted per benchmark and under the comparison policy below, rather than as a single controlled total order.

4.1 Evaluation Setup and Comparison Policy

Benchmarks and Metrics. We evaluate XYZ-Aquila on seven challenging agentic search benchmarks: BrowseComp, BrowseComp-ZH, DeepSearchQA, GAIA, LiveBrowseComp, Humanity's Last Exam (HLE), and WideSearch. Together, they cover multilingual web browsing, dynamic information retrieval, long-horizon evidence acquisition, broad information seeking, evidence aggregation, and knowledge-intensive reasoning

We report accuracy on BrowseComp, BrowseComp-ZH, GAIA, LiveBrowseComp, and HLE; F1 on DeepSearchQA; and Item F1 Max@4 on WideSearch, following the corresponding benchmark protocols. All values in the tables are percentages. The small-scale comparison covers all seven benchmarks; the current large-scale comparator set covers six, so GAIA is not included in Table 3. Because LiveBrowseComp depends on the web state at evaluation time, its margins should be interpreted as time-specific observations rather than permanent differences.

Models and Evaluation Protocol. We evaluate two members of the XYZ-Aquila family: **XYZ-Aquila-mini**, based on the Qwen3.6-35B-A3B backbone, and **XYZ-Aquila-pro**, based on the Qwen3.5-397B-A17B backbone. We compare them against representative open-weight search agents across different model scales, as well as frontier-scale or proprietary systems when publicly available benchmark results are available.

We evaluate all agents using the search-agent evaluation infrastructure introduced in Section 3. The evaluation follows a ReAct-style interaction protocol, where agents iteratively perform tool calls, receive observations from the external environment, and produce final answers after evidence gathering and verification. All evaluation configurations, including retrieval, webpage processing, execution environment, and interaction settings, follow the setup described in Section 3. We use a maximum context length of 256K tokens.

Comparison provenance. Results marked with † are reproduced by us under the common harness and therefore provide the most directly controlled comparisons in the tables. Unmarked baseline values are collected from publicly available reports or benchmark submissions and may differ in tool stack, retrieval backend, context policy, judge, normalization, evaluation date, or other implementation details. We consequently describe such comparisons as *reported benchmark-level comparisons*. Missing entries are denoted by "–" and mean that a result was unavailable; they are not zero scores. Because benchmark coverage is incomplete and the reported metrics are not homogeneous, we do not compute a single aggregate score across benchmarks.

Model Name	BrowseComp	BrowseComp-ZH	DeepSearchQA	GAIA	LiveBrowseComp	HLE	WideSearch
Agents-A1	<u>75.5</u>	–	–	<u>96.0</u>	29.6†	<u>47.6</u>	–
Nex-N2-mini	74.1	79.6†	<u>87.2</u>†	–	<u>41.4</u>†	37.1†	62.0
apodex-mini	71.5	<u>80.6</u>	82.2	–	32.8†	46.8	–
MiroThinker 1.7 mini	67.9	–	–	80.3	34.9†	36.4	<u>73.3</u>†
XYZ-Aquila-mini	**78.8**	**82.9**	**89.5**	**97.1**	**48.7**	**51.1**	**80.8**

Table 2: Benchmark results for small-scale (<40B) open-weight search agents, reported as percentages. For DeepSearchQA, we report F1; for WideSearch, we report Item F1 Max@4. Missing entries indicate unavailable results. Within this table, the best result in each benchmark is bolded, and the second-best result is underlined. Results marked with † are reproduced by us using our evaluation setup; unmarked baseline results are collected from publicly available reports or benchmark submissions.

Within each explicitly defined comparison group, the best value in a column isbolded and the second-best is underlined.

4.2 External Benchmark Results

XYZ-Aquila-mini: matched-scale leadership. Table 2 compares XYZ-Aquila-mini with representative open-weight search agents below 40B parameters, including Apodex-mini, Agents-A1, MiroThinker 1.7 mini, and Nex-N2-mini. XYZ-Aquila-mini achieves the highest reported scores across all seven benchmarks, reaching 78.8%, 82.9%, 89.5%, 97.1%, 48.7%, 51.1%, and 80.8% on BrowseComp, BrowseComp-ZH, DeepSearchQA, GAIA, LiveBrowseComp, HLE, and WideSearch, respectively. No single baseline reports results on all seven benchmarks, so this statement is a column-wise comparison rather than a complete seven-task head-to-head aggregate. The largest margins over the next reported score in the table occur on WideSearch (+7.5 points) and LiveBrowseComp (+7.3 points); in both cases, the runner-up score was reproduced under our harness. The results also span English and Chinese browsing, general tool-assisted reasoning, dynamic-web retrieval, and broad evidence aggregation, indicating that the advantage is not confined to one benchmark family.

XYZ-Aquila-pro: open-weight leadership and cross-scale competitiveness. Table 3 first compares open-weight agents below 400B parameters, then provides a contextual comparison with frontier-scale or proprietary systems for which public scores are available. Within the open-weight group, XYZ-Aquila-pro achieves the highest reported value in all six columns: 84.8 on BrowseComp, 85.1 on BrowseComp-ZH, 92.5 F1 on DeepSearchQA, 53.7 on LiveBrowseComp, 53.3 on HLE, and 81.2 Item F1 Max@4 on WideSearch. Its largest within-group lead is on WideSearch (+5.6 points), followed by LiveBrowseComp and HLE (+3.3 points each).

Across all systems listed in Table 3, XYZ-Aquila-pro has the highest reported score on BrowseComp-ZH, LiveBrowseComp, and WideSearch. On BrowseComp, only apodex-h1 reports a higher value; on DeepSearchQA, XYZ-Aquila-pro ties Kimi-K2.6 at 92.5 F1 and trails apodex-h1; on HLE, several larger systems remain ahead. These cross-scale comparisons locate XYZ-Aquila-pro relative to the reported frontier, but they are not uniformly controlled because the unmarked values come from different evaluation stacks and dates. We therefore treat the sub-400B open-weight comparison as the primary matched-scale result and the second group as broader context.

4.3 Trace-Level Tool-Use Patterns

Beyond aggregate benchmark scores, we analyze the behavioral characteristics of XYZ-Aquila through execution traces. The structured traces generated during evaluation provide insights into how agents allocate tool usage across different search environments. Figure 5 summarizes

Model Name	BrowseComp	BrowseComp-ZH	DeepSearchQA	LiveBrowseComp	HLE	WideSearch
Open-weight Agents (<400B Parameters)						
Nex-N2-Pro	<u>83.7</u>	79.6†	<u>92.3</u>†	<u>50.4</u>†	<u>50.0</u>†	<u>75.6</u>
MiroThinker 1.7	74.0	75.3	–	34.1†	42.9	–
apodex-1.0	75.5	<u>82.6</u>	84.6	–	49.0	–
XYZ-Aquila-pro	**84.8**	**85.1**	**92.5**	**53.7**	**53.3**	**81.2**
Frontier-scale and Proprietary Agents						
apodex-h1	**90.3**	**84.1**	**94.4**	–	**60.8**	–
DeepSeek-V4-Pro (1.6T, Max) [34]	83.4	–	–	**38.3**	–	–
Kimi-K2.6 (1T) [35]	83.2	–	<u>92.5</u>	<u>31.7</u>	<u>55.5</u>	**80.8**
Claude Opus 4.7	79.3	–	89.1	–	54.7	–
GPT-5.5 xhigh	<u>84.4</u>	–	–	–	52.2	–
Qwen3.6-Plus	–	–	–	–	–	<u>74.3</u>

Table 3: Benchmark results for large-scale search agents, reported as percentages. Models are divided into two comparison groups: open-weight agents with publicly available weights and frontier-scale or proprietary agents. For DeepSearchQA, we report F1; for WideSearch, we report Item F1 Max@4. Missing entries indicate unavailable results. Within each comparison group, the best result in each benchmark is bolded and the second-best result is underlined. Results marked with † are reproduced by us using our evaluation setup; unmarked baseline results are collected from publicly available reports or benchmark submissions.

the tool-use composition across evaluated benchmarks, including web search, webpage extraction, and Python execution.

The benchmark-level distributions differ substantially. Web search accounts for 78% of sampled calls on BrowseComp and 84% on both BrowseComp-ZH and LiveBrowseComp, consistent with tasks dominated by broad retrieval and source discovery. DeepSearchQA and GAIA devote 54% and 55% of their calls, respectively, to webpage extraction and Python execution combined, reflecting heavier evidence processing and intermediate computation after retrieval. WideSearch uses a mixture of search (61%) and scrape/extract (39%) with no Python calls in the sampled traces, while HLE exhibits an intermediate profile with 64% search, 14% extraction, and 22% Python.

These results suggest that effective search agents require adaptive tool-use strategies rather than a fixed interaction pattern. Different benchmarks stress different components of the agentic search pipeline, and strong performance depends on selecting appropriate tools for the underlying information structure.

Taken together, the external results establish strong matched-scale performance for both XYZ-Aquila systems and cross-scale competitiveness for XYZ-Aquila-pro, while the trace statistics characterize how the resulting agents distribute tool use across task types. These findings evaluate the final systems as deployed under the stated protocol; isolating the causal contribution of individual AI4AI interventions requires component-level ablations beyond the scope of this section.

5 Discussion, Limitations, and Future Work

The results in Section 4 support a bounded system-optimization claim: under a human-defined contract, the AI4AI process produced two strong search-agent configurations, and those configurations performed well on external benchmarks withheld from routine optimization. They do not isolate model scale or any single intervention as the cause of the gains, establish a universal ordering across agents, or justify claims beyond the released systems and evaluation conditions. The appropriate unit of analysis is therefore the complete released system—backbone, data pipeline, post-training recipe, runtime, context policy, replay infrastructure, and tool interface—evaluated under the stated harness and evaluation date. We next discuss the conditions under which such a process is trustworthy, where it may transfer, and which empirical questions remain

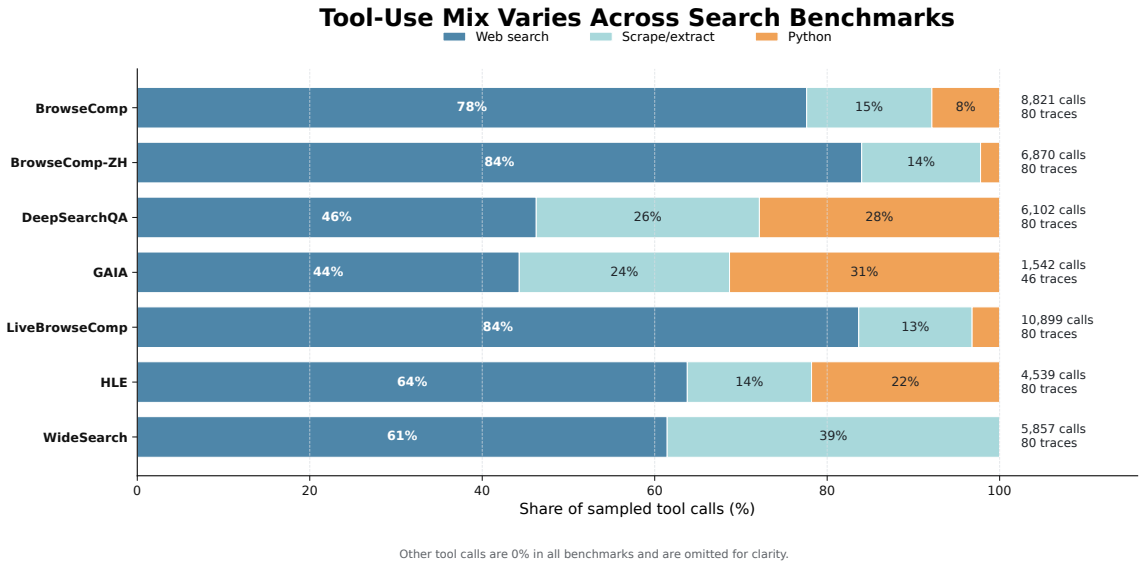

Figure 5: Tool-use composition varies substantially across search benchmarks. Each bar shows the proportion of sampled tool calls assigned to web search, webpage extraction, and Python execution. Retrieval-oriented benchmarks rely primarily on search operations, while reasoning-intensive benchmarks allocate more interactions to extraction and computation, reflecting their different information processing requirements.

open.

5.1 Auditability and Trust

Reconstructable optimization. The credibility of bounded recursive improvement depends on whether each state transition can be reconstructed. For every candidate, the cycle record links the active contract and incumbent to the proposed intervention, its rationale, candidate artifacts, data lineage, code and prompt changes, model and tool versions, training and tool-call traces, resource use, evaluator outputs, gate decision, reviewer input, and rollback state. Rejected interventions are retained alongside accepted ones. This negative evidence is essential: without it, the record describes only the winning configuration and cannot reveal the search path, repeated failure modes, or selection pressure that produced it.

Evaluator governance. Auditability also requires role separation. An observed gain should be attributable to the candidate system rather than to a more permissive interface or a favorable scoring change. Changes to judge prompts, normalization rules, or benchmark-specific scoring must therefore follow a separate, versioned validation path, be checked against human-labeled examples, and trigger re-baselining before they can govern adoption. Auditability does not forbid evaluator maintenance; it prevents evaluator changes from being folded into the same decision they would affect.

Benchmark isolation. The private development benchmark is accessed only through isolated gate evidence. Its answer keys and labels remain hidden from the optimizer and rollout agent; data and trajectory pipelines are screened for overlap and near-duplicates, with suspected leakage escalated to human review; and each candidate receives at most one gate query at the cycle boundary. The external suite is excluded from the optimizer's observation history and shared memory and is used only for final or explicitly authorized assessment. These controls make external evaluation a test of transfer rather than another adaptive optimization signal. They do

not make heterogeneous public comparisons fully controlled: results should still be interpreted per benchmark and under the stated harness, tool stack, judge version, and evaluation date.

5.2 Broader Applicability

Conditions for transfer. Transfer beyond Deep Search is conditional rather than automatic. The loop is most applicable when the target behavior admits a versioned, domain-relevant metric; decisive labels or tests can be isolated from the optimizer; actions and system states are replayable; and humans can specify meaningful permission, resource, safety, and acceptance boundaries. Without these properties, AI-assisted iteration may still accelerate development, but the resulting recursion is harder to attribute and govern.

Coding agents. Coding agents [36, 37, 38] are a natural next test bed. Their traces expose repository conventions, dependency failures, test outputs, review comments, and patch histories that can be converted into data filters, verifier checks, trajectory curricula, and regression tests. The central risk is reward hacking: an agent may overfit visible tests, exploit benchmark shortcuts, or produce a brittle patch that passes narrow checks while reducing maintainability [11, 39]. A bounded loop is useful precisely because hidden tests, broader quality checks, and human review can remain outside the optimizer's direct control.

Specialized domains. In specialized domains such as science [40] and medicine [41], the same mechanism could support local adaptation to documents, tools, workflows, and recurring failure modes. Such use should be framed as expert-supervised adaptation rather than autonomous domain expansion: domain validity, safety, privacy, provenance, and deployment authority must be represented in the contract and acceptance gate, with human experts retaining control over consequential decisions.

5.3 Limitations and Future Work

Causal attribution and path dependence. The present study evaluates the endpoint systems, not a complete causal decomposition of the optimization trajectory. Because later interventions build on earlier ones, their effects can be path-dependent and non-additive; a simple leave-one-component-out analysis may therefore miss important interactions. Future work should combine counterfactual replay from common checkpoints, controlled retraining under fixed budgets, and targeted factorial ablations of high-interaction components such as data and SFT, backbone and training recipe, and runtime and context policy. Reporting both accepted and rejected interventions under these controls would clarify marginal effects, interaction effects, and transfer across backbones.

Adaptive overfitting and evaluator dependence. Evaluator isolation reduces but does not eliminate adaptive overfitting to the private development distribution. Aggregate scores, failure splits, and approved diagnostics still form a repeated feedback channel across cycles, and leakage screening cannot rule out all semantic overlap. Future work should use rotating private splits, periodically refreshed hidden holdouts, pre-specified gate criteria, and scheduled transfer audits to measure whether improvements persist outside the development distribution.

Comparator and temporal validity. Only a subset of comparator results is reproduced under the common harness; the remaining values come from heterogeneous reports, tools, judges, and dates. Live web benchmarks introduce an additional source of temporal variation, and the current tables do not fully characterize run-to-run or judge sensitivity. Future evaluation

should expand common-harness reproduction, report uncertainty across repeated runs and evaluation dates, archive query-time artifacts where benchmark rules permit, and include sensitivity analyses for judges and normalization rules.

Cost, latency, and supervision. The report does not yet provide end-to-end accounting of development compute, inference latency, tool cost, evaluator cost, human review time, or memory and audit overhead. The current efficiency evidence is therefore primarily parameter- and score-based. Future work should report score–cost–latency–reliability Pareto curves, together with escalation rates and the distribution of human effort, to determine when the bounded loop is operationally advantageous.

RL and transfer beyond search. The answer-conditioned process-reward design passed low-cost screening but did not complete full RL training and gate evaluation, so the study does not establish an RL contribution to the released systems. A controlled RL study should measure performance, stability, reward hacking, and cost across the full suite before adoption. More broadly, empirical validation is limited to Deep Search and two backbones. Coding agents provide the next direct test of transfer, while scientific and medical applications require stricter domain-specific gates and expert validation.

Taken together, the current evidence establishes that a bounded, auditable loop can produce strong search agents under a specific contract. It does not yet establish which components are causally necessary, whether the development process is resource-efficient end to end, or how reliably the recipe transfers across domains.

6 Conclusion

We introduced bounded-exploration AI4AI as a governed method for system-level agent improvement. Humans define and version the optimization contract; AI agents diagnose failures and implement scoped interventions; an isolated evaluator supplies evidence to the acceptance gate; and accepted and rejected outcomes are retained with their provenance. This separation of roles makes the loop recursive without making it open-ended and converts both successful and failed experiments into reusable, auditable experience.

Applied to Deep Search, the process produced XYZ-Aquila-mini and XYZ-Aquila-pro. The released systems combine selected backbones with graph-grounded, tool-reachable task construction, state-faithful supervised fine-tuning, and a fixed agent-facing tool contract with explicit context-management and replay policies. The screened reinforcement-learning proposal is not part of either release because it did not complete the full training-and-gate path. On the external suite withheld from routine optimization, XYZ-Aquila-mini leads every column of the seven-benchmark sub-40B open-weight comparison, while XYZ-Aquila-pro leads every column of the six-benchmark sub-400B open-weight comparison. XYZ-Aquila-pro also records the highest reported score among all listed systems on BrowseComp-ZH, LiveBrowseComp, and WideSearch. Because some comparator values come from heterogeneous public reports, these findings support strong matched-scale and benchmark-level performance rather than a universal ranking.

Taken together, the evidence supports a bounded system-optimization claim: under a human-defined contract, coordinated changes across data, learning, runtime, context management, and infrastructure produced strong search agents while keeping evaluation isolated and the optimization path reconstructable. It does not identify a single causal component, establish end-to-end operational efficiency, or demonstrate transfer beyond search. Controlled ablations, cost and latency accounting, rotating holdouts, full RL validation, and evaluation in domains such as coding are therefore the next steps. The broader lesson is that agentic progress depends not

only on scaling model checkpoints, but also on engineering the improvement process itself to be measurable, bounded, cumulative, and accountable.

References

[1] Reiichiro Nakano et al. WebGPT: Browser-assisted question-answering with human feedback, 2021. arXiv:2112.09332. https://arxiv.org/abs/2112.09332.

[2] Shunyu Yao et al. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2210.03629. https://arxiv.org/abs/2210.03629.

[3] Timo Schick et al. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2302.04761. https://arxiv.org/abs/2302.04761.

[4] Grégoire Mialon et al. GAIA: a benchmark for General AI Assistants. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2311.12983. https://arxiv.org/abs/2311.12983.

[5] Chris Lu et al. The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery, 2024. arXiv:2408.06292. https://arxiv.org/abs/2408.06292.

[6] Samuel Schmidgall et al. Agent Laboratory: Using LLM Agents as Research Assistants, 2025. arXiv:2501.04227. https://arxiv.org/abs/2501.04227.

[7] Jun Shern Chan et al. MLE-bench: Evaluating Machine Learning Agents on Machine Learning Engineering. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.07095. https://arxiv.org/abs/2410.07095.

[8] Zhengyao Jiang et al. AIDE: AI-Driven Exploration in the Space of Code, 2025. arXiv:2502.13138. https://arxiv.org/abs/2502.13138.

[9] Zexi Liu et al. ML-Master: Towards AI-for-AI via Integration of Exploration and Reasoning, 2025. arXiv:2506.16499. https://arxiv.org/abs/2506.16499.

[10] Wanyi Chen, Xiao Yang, Xu Yang, Tianming Sha, Qizheng Li, Zhuo Wang, Bowen Xian, Fang Kong, Weiqing Liu, and Jiang Bian. Agent^2 RL-Bench: Can LLM Agents Engineer Agentic RL Post-Training?, 2026.

[11] Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and Characterizing Reward Hacking, 2022. arXiv:2209.13085. https://arxiv.org/abs/2209.13085.

[12] Jason Wei et al. BrowseComp: A Simple Yet Challenging Benchmark for Browsing Agents, 2025. arXiv:2504.12516. https://arxiv.org/abs/2504.12516.

[13] Peilin Zhou et al. BrowseComp-ZH: Benchmarking Web Browsing Ability of Large Language Models in Chinese, 2025. arXiv:2504.19314. https://arxiv.org/abs/2504.19314.

[14] Nikita Gupta et al. DeepSearchQA: Bridging the Comprehensiveness Gap for Deep Research Agents, 2026. arXiv:2601.20975. https://arxiv.org/abs/2601.20975.

[15] HuiMing Fan et al. LiveBrowseComp: Are Search Agents Searching, or Just Verifying What They Already Know?, 2026. arXiv:2605.28721. https://arxiv.org/abs/2605.28721.

[16] Long Phan et al. Humanity's Last Exam, 2025. arXiv:2501.14249. https://arxiv.org/abs/2501.14249.

[17] Ryan Wong et al. WideSearch: Benchmarking Agentic Broad Info-Seeking, 2025. arXiv:2508.07999. https://arxiv.org/abs/2508.07999.

[18] MiroMind Team. MiroThinker-1.7 & H1: Towards Heavy-Duty Research Agents via Verification, 2026. arXiv:2603.15726. https://arxiv.org/abs/2603.15726.

[19] Lei Bai et al. Scaling the Horizon, Not the Parameters: Reaching Trillion-Parameter Performance with a 35B Agent, 2026. arXiv:2606.30616. https://arxiv.org/abs/2606.30616.

[20] NexAGI. Nex-N2: An Open-source Deep Research Agent. GitHub repository, 2026. https://github.com/nex-agi/Nex-N2.

[21] Apodex. Apodex-1.0-mini. Hugging Face model card, 2026. https://huggingface.co/apodex/Apodex-1.0-mini.

[22] ApodexAI. AgentHarness. GitHub repository, 2026. https://github.com/ApodexAI/AgentHarness.

[23] Xu Yang, Xiao Yang, Shikai Fang, Yifei Zhang, Jian Wang, Bowen Xian, Qizheng Li, Jingyuan Li, Minrui Xu, Yuante Li, et al. R&d-agent: An llm-agent framework towards autonomous data science. *arXiv preprint arXiv:2505.14738*, 2025.

[24] Yifei Zhang, Xu Yang, Xiao Yang, Bowen Xian, Qizheng Li, Shikai Fang, Jingyuan Li, Jian Wang, Minrui Xu, Yuge Zhang, et al. Reasoning as Gradient: Scaling MLE Agents Beyond Tree Search, 2026. arXiv:2603.01692. https://arxiv.org/abs/2603.01692.

[25] Qizheng Li, Yifei Zhang, Xiao Yang, Xu Yang, Zhuo Wang, Weiqing Liu, and Jiang Bian. Ft-dojo: Towards autonomous llm fine-tuning with language agents. *arXiv preprint arXiv:2603.01712*, 2026.

[26] Long Ouyang et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. arXiv:2203.02155. https://arxiv.org/abs/2203.02155.

[27] Yuntao Bai et al. Constitutional AI: Harmlessness from AI Feedback, 2022. arXiv:2212.08073. https://arxiv.org/abs/2212.08073.

[28] Rafael Rafailov et al. Direct Preference Optimization: Your Language Model is Secretly a Reward Model, 2023. arXiv:2305.18290. https://arxiv.org/abs/2305.18290.

[29] Noah Shinn et al. Reflexion: Language Agents with Verbal Reinforcement Learning, 2023. arXiv:2303.11366. https://arxiv.org/abs/2303.11366.

[30] MiniMax Research. Forge: Scalable agent RL framework and algorithm. https://www.minimax.io/blog/forge-scalable-agent-rl-en-1779896141, 2026.

[31] Tao Zewei and Huang Yunpeng. Magiattention: A distributed attention towards linear scalability for ultra-long context, heterogeneous mask training. https://github.com/SandAI-org/MagiAttention/, 2025.

[32] Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, et al. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv preprint arXiv:2506.14245.

[33] Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-Group Policy Optimization for LLM Agent Training. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pages 46375–46408, 2025. https://proceedings.neurips.cc/paper_files/paper/2025/hash/420c9f777c0b4f78d515e53cf74d58b2-Abstract-Conference.html.

[34] DeepSeek-AI. DeepSeek-V4: Towards Highly Efficient Million-Token Context Intelligence. Technical report, 2026. https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro/blob/main/DeepSeek_V4.pdf.

[35] Moonshot AI. Kimi K2.6. Hugging Face model card, 2026. https://huggingface.co/moonshotai/Kimi-K2.6.

[36] John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

[37] Anthropic. Claude 3.7 sonnet and claude code. https://www.anthropic.com/news/claude-3-7-sonnet. Introduces Claude Code as an agentic command-line coding tool.

[38] OpenAI. Introducing codex. https://openai.com/index/introducing-codex/. Cloud-based software engineering agent powered by codex-1.

[39] Naman Jain. Reward hacking is swamping model intelligence gains. https://cursor.com/blog/reward-hacking-coding-benchmarks. Cursor Blog.

[40] Kyle Swanson, Wesley Wu, Nash L Bulaong, John E Pak, and James Zou. The virtual lab of ai agents designs new sars-cov-2 nanobodies. *Nature*, 646(8085):716–723, 2025.

[41] Dyke Ferber, Lars Hilgers, Christiane Höper, Benedict Kinny-Köster, Jan-Niklas Eckardt, Katharina Egger-Heidrich, Marius Bill, Martin MK Schneider, Jan Clusmann, Lejla Kadric, et al. Towards autonomous medical artificial intelligence agents. *Nature*, pages 1–10, 2026.

A Authors

The following authors are listed in alphabetical order by last name. An asterisk denotes the corresponding author.

Jiang Bian*	Wanyi Chen	Yuwen Du
Jingjing Fu	Xiaofan Gui	Tianyu He
Yuhao Huang	Yuting Jiang	Junjie Li
Qizheng Li	Sijia Li	Yuhuan Liu
Xufang Luo	Dan Qiao	Tianming Sha
Lei Song	Shizhao Sun	Yang Tian
Hui Wang	Jinyu Wang	Peidong Wang
Rui Wang	Yang Wang	Xumeng Wen
Yueliang Xie	Shuotao Xu	Mao Yang
Xiao Yang	Xu Yang	Ziyue Yang
Rui Ye	Yang Ye	Xiaohan Yi
Shenghai Yuan	Yang Yue	Yifei Zhang
Haitian Zhong	Liao Zhu	

*Corresponding author.

B Reproducibility Details

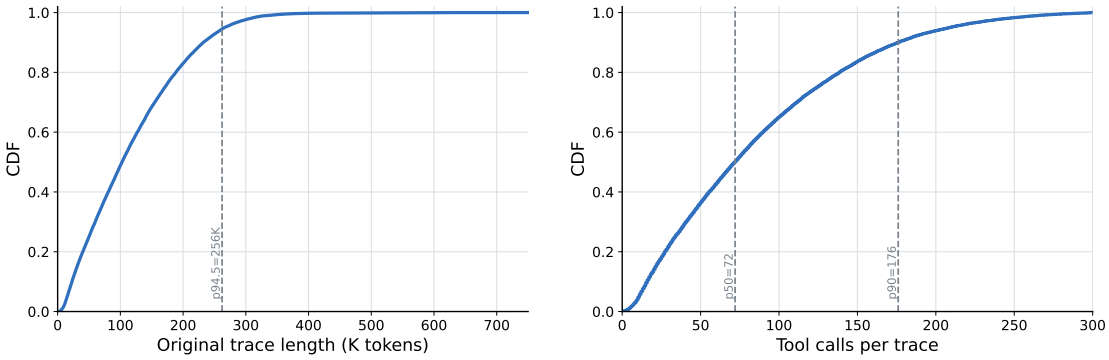

Figure 6: Empirical cumulative distributions of complete rollout-trace token length (left) and tool-call count (right).

Table 4: SFT training configuration for XYZ-Aquila-mini and XYZ-Aquila-pro.

Configuration item	XYZ-Aquila-mini	XYZ-Aquila-pro
Learning rate	2×10^{-5}	1×10^{-5}
Scheduler	Cosine decay	Cosine decay
Minimum learning rate	2×10^{-6}	1×10^{-6}
Adam β_1	0.9	0.9
Adam β_2	0.98	0.98
Loss aggregation	Per-token loss	Per-token loss
MoE auxiliary loss	False	False
Warmup ratio	0.05	0.05
Global batch size	32	64
Packing length	262,144	262,144
Optimizer	AdamW	AdamW
Weight decay	0.1	0.1

Full-rollout SFT trace distribution. Figure 6 summarizes the complete traces produced by the pipeline in Section 3.2.2. Median trace length and tool use are 102.7K tokens and 72 calls; the 90th percentiles are 232.1K tokens and 176 calls, with the tool-use tail approaching the 300-turn horizon. Complete length is measured over the append-only log, whereas each rollout step exposes at most 256K tokens to the model. Keeping recent tool interactions and compressing or summarizing older context lets difficult searches continue after an uncompressed history would fill that window. Consequently, 5.5% of retained correct traces exceed 256K in total logged length, and the longest reaches 749.4K, while every generation step remains within the 256K limit. During SFT conversion, the recorded context-management events are replayed to recover the exact visible context for each turn, yielding aligned, bounded training samples. This preserves longer chains of search, evidence integration, and recovery without changing the model's context limit.

SFT training configuration. Table 4 reports the optimization, batching, and packing settings used for XYZ-Aquila-mini and XYZ-Aquila-pro.