ARK: A Human-Steered Research Harness for AI-Assisted Scientific Discovery

Jihao Xin [*1] **Yangzhixin Luo** [*1] **Muhammad Bilal** [1] **Marco Canini** [1]

1. Introduction

Automating research processes to accelerate scientific discovery are emerging. Existing works (The AI Scientist (Lu et al., 2026), AutoResearch (Karpathy, 2026), ARIS (Yang et al., 2026)) demonstrate the feasibility of AI scientists to automate research in different domains. However, existing works pursue a full automation paradigm without human supervision, which is prone to uncontrolled results, e.g., intention drift and hallucination (Miyai et al., 2026).

We argue that the right paradigm is the *co-scientist* (Gottweis et al., 2026): a system that offloads downstream implementation, execution, and writing labor to AI agents, while humans *steer the science* with full creative and strategic control. This is captured in our design philosophy: **Full Automation with Full Control**.

We present **ARK** (**A**utomatic **R**esearch **K**it), a co-scientist framework that automates the full idea-to-paper pipeline while keeping the researcher in control at every stage. While problem-solving systems focus on generating hypotheses or algorithms, ARK starts from a solid idea and carries the downstream path from execution to manuscript. We design ARK as a model-agnostic multi-agent system whose seven agents and three-phase pipeline turn a research idea end-to-end into a *publication-ready* paper.

We propose ARK in the form of **Research as a Service** (RaaS) that turns research from a local, manual endeavor into a web service. To use ARK, the human scientist only needs to provide the research idea, then launch it in one click without any local setup. Each project runs in an isolated sandbox with dedicated dependencies, filesystem, and credentials to prevent cross-project contamination, and supports three interchangeable backends: *Local*, *Cluster*, and *Cloud*. Users can monitor the research progress on the website or via Telegram, and inject instructions into the system at any time.

ARK is live at idea2paper.org, with source code at github.com/kaust-ark/ARK.

[*]Equal contribution [1]KAUST. Correspondence to: Marco Canini <marco@kaust.edu.sa>.

Accepted at ICML 2026 AI for Science Workshop.

2. System Design

ARK adopts a three-phase pipeline to turn a research idea into a paper. Figure 1 illustrates ARK's architecture.

Phase 1: Initialization & Research. A user submits a research idea, then the *Researcher*[1] parses the input and produces: (a) a research *query* for Deep Research, and (b) a project *briefing* that is used to keep downstream agents aligned with user intent. Deep Research conducts a comprehensive literature survey, identifies research gaps, proposes methodology, and designs a concrete experimental protocol, specifying the required tools, datasets, and compute resources. Combining the Deep Research output with the project briefing, ARK performs agent specialization, customizing agent prompts with project-specific instructions. ARK then retrieves project-specific skills from our curated skill library, equipping the system with external knowledge.

Phase 2: Iterative Development. The *Planner* turns the project briefing and Deep Research plan into an experimental plan, and the *Experimenter* executes it on a configurable compute backend. The *Planner* then evaluates completeness and loops back to refine or extend experiments until the evidence is sufficient. Once the loop converges, the *Writer* produces an initial manuscript draft.

Phase 3: Iterative Review. ARK mirrors academic peer review, covering innovation, presentation, technical depth, and experimental rigor. (1) ARK compiles LaTeX to PDF; (2) the *Reviewer* scores the paper, identifies major/minor issues, and gives accept/reject recommendations; (3) the *Planner* analyzes each issue and emits prioritized action plans; (4) ARK dispatches independent tasks for parallel execution; (5) ARK validates the changes and adjusts the layout. The loop repeats until the score exceeds an acceptance threshold or a maximum iteration count is reached.

Implementation Details

- **Specialized agents.** *Researcher*, *Planner*, *Experimenter*, *Coder*, *Writer*, *Reviewer*, and *Meta-Debugger*, with responsibilities detailed in Appendix A.
- **Memory System.** A *Memory System* persists cross-

[1] denotes an ARK agent.

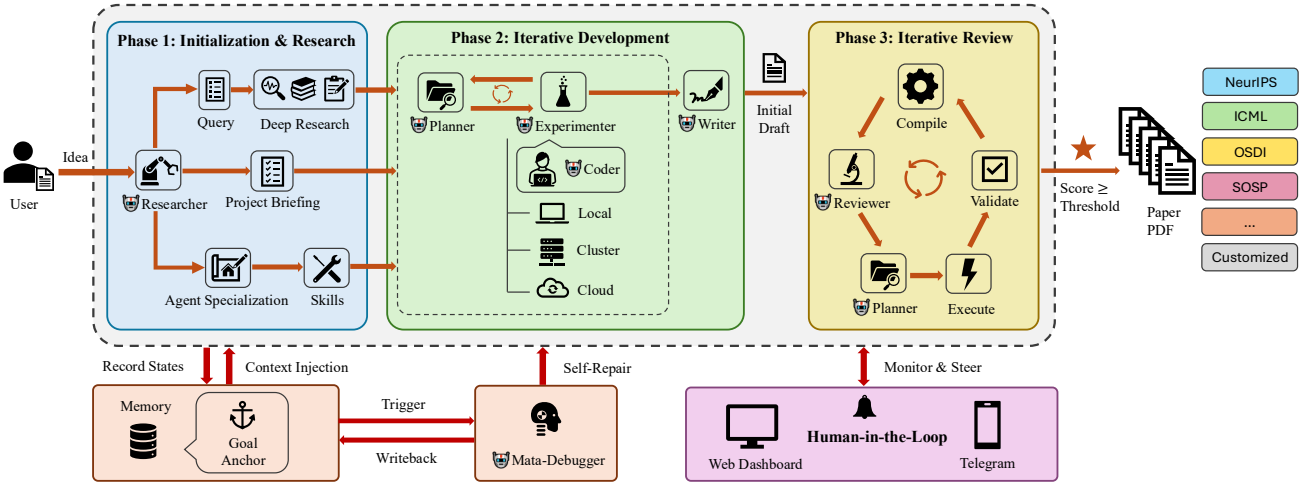

Figure 1. Overview of the ARK framework.

iteration state, tracking score history, detecting stagnation, and triggering the *Meta-Debugger*. A persistent *Goal Anchor* of the user's intent is re-injected into every agent call to prevent drift across long iteration sequences.

- **Deployment.** ARK web is developed with FastAPI. The backend server runs a daemon, every project launch spawns an orchestration process and runs under its own sandbox. The system data is stored in a relational database, and project data is stored as files.
- **Security & traffic control.** ARK server is protected with edge-side authentication and traffic control to ensure secure service. Sensitive data is Fernet encrypted.
- **Token & cost transparency.** Per-call tokens and costs are parsed, aggregated and displayed in real time.
- **Layout Control.** ARK maintains LaTeX templates for major venues, compiles drafts to PDF page images for visual-grounded review, and scans .tex, .aux, and .sty files to eliminate errors. For page limits, ARK inserts source markers and reads .aux artifacts to estimate section lengths and revise the manuscript to fit the venue budget.
- **Figure generation.** Diagrams are generated by PaperBanana (Zhu et al., 2026), and statistic plots are generated under venue-aware canvas geometry by Matplotlib.

3. Key Technical Innovations

Agent Specialization. Our experiments show that fixed agent prompts cannot suit all research tasks. Therefore, we propose a *Template-Specialization* design. ARK abstracts the generic logic of an agent into an *Agent Template*. After Deep Research, ARK retrieves the templates and specializes them for each project by injecting project-specific knowledge into the template, covering domain-specific knowledge and experiment details.

Hallucination Validation. To reduce LLM hallucinations, ARK wraps generation with *deterministic validation* where external evidence is available. For example, an LLM might cite non-existent papers, so in line with retrieval-grounded scientific QA (Lálá et al., 2023), ARK integrates API-first BibTeX retrieval and content–claim alignment: it retrieves references from trusted APIs (e.g., DBLP, CrossRef, arXiv) and lets the *Writer* cite them only where the content supports the claim. For experimental claims, ARK stores run outputs in canonical artifacts and constrains the *Writer* to extract reported numbers rather than free-form generation.

Retrieval-Augmented Skills. ARK is designed to be general and extensible. We curated a skill library from public repositories (K-Dense Inc., 2026) to support research across domains. During initialization, the *Researcher* retrieves relevant skills from the library and installs them into the project's sandbox, where they become available to all downstream agents. The current library focuses on computation-based experiments, with physical-world extensions (laboratory experiments, robotics) under development.

Human-in-the-Loop Control. Full automation will drift away from the user's intent. ARK keeps the researcher in full control through three mechanisms: *Real-time Monitoring*: ARK reports progress, cost, and intermediate outputs in real time via the website and Telegram. *Key-point Notification*: At decision points such as ambiguous instructions or budget approvals, ARK messages the user with suggested actions and asks for confirmation. *Anytime Intervention*: ARK accepts instructions from the user via Telegram or the website at any time and updates the project state accordingly.

4. Evaluation

An ARK-powered paper has been peer-reviewed and published in the ACM Digital Library. To quantify paper-artifact quality, we use three reviewer systems—the ARK reviewer, the Sakana AI Scientist reviewer, and the CSPaper.org reviewer—with setup and scores reported in Appendix B. All three reviewers rank ARK $>$ ARIS $\geq$ Sakana.

References

Anthropic. Claude Code: AI coding agent, terminal, IDE. https://code.claude.com/docs, 2025. Accessed: April 2026.

Gemini Team. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Gottweis, J., Weng, W.-H., Daryin, A., Tu, T., Sirkovic, P., Myaskovsky, A., Glowaty, G., Weissenberger, F., Orlandi, A., Popovici, D., et al. Accelerating scientific discovery with co-scientist. *Nature*, pp. 1–3, 2026.

K-Dense Inc. Scientific agent skills: A comprehensive collection of scientific tools for ai agents. https://github.com/K-Dense-AI/scientific-agent-skills, 2026. GitHub repository.

Karpathy, A. AutoResearch: AI agents running research on single-GPU nanochat training automatically. https://github.com/karpathy/autoresearch, 2026. GitHub repository.

Lálá, J., O'Donoghue, O., Shtedritski, A., Cox, S., Rodriques, S. G., and White, A. D. PaperQA: Retrieval-augmented generative agent for scientific research. *arXiv preprint arXiv:2312.07559*, 2023.

Lu, C., Lu, C., Lange, R. T., Yamada, Y., Hu, S., Foerster, J., Ha, D., and Clune, J. Towards end-to-end automation of AI research. *Nature*, 651(8107):914–919, 2026.

Miyai, A., Toyooka, M., Otonari, T., Zhao, Z., and Aizawa, K. Jr. AI Scientist and its risk report: Autonomous scientific exploration from a baseline paper. *arXiv preprint arXiv:2511.04583*, 2026.

OpenAI. Codex: AI coding partner from OpenAI. https://openai.com/codex/, 2025. Accessed: April 2026.

Steinberger, P. and the OpenClaw community. OpenClaw: Personal AI assistant. https://github.com/openclaw/openclaw, 2026. GitHub repository.

Yang, R., Li, Y., and Li, S. ARIS: Autonomous research via adversarial multi-agent collaboration. *arXiv preprint arXiv:2605.03042*, 2026.

Zhu, D., Meng, R., Song, Y., Wei, X., Li, S., Pfister, T., and Yoon, J. PaperBanana: Automating academic illustration for AI scientists. *arXiv preprint arXiv:2601.23265*, 2026.

A. Specialized Agents

ARK orchestrates seven specialized agents: *Researcher*, *Planner*, *Experimenter*, *Coder*, *Writer*, *Reviewer*, and *Meta-Debugger*. They collaborate through shared project states (manuscript source, experimental results, score history, and Goal Anchor). All inter-agent communication is mediated through structured YAML action plans, ensuring reproducibility and auditability. Each agent runs on a pluggable LLM runtime: Claude Code (Anthropic, 2025) by default, with OpenAI Codex (OpenAI, 2025) and Google Gemini (Gemini Team, 2023) as drop-in alternatives, so other backends can be swapped in without changing the orchestration layer.

- *Researcher*: Parses the user's research idea and produces the refactored query for Deep Research and the project briefing that anchors all subsequent phases.
- *Planner*: Orchestrates the pipeline by turning research goals and review feedback into prioritized action plans, assigning each task to the appropriate agent.
- *Experimenter*: Designs experiments, submits compute jobs (Slurm/cloud/local), analyzes results.
- *Coder*: Implements and debugs experiment and analysis code.
- *Writer*: Writes and revises the paper in LaTeX.
- *Reviewer*: Scores papers (1–10) against venue standards using rendered page images *and* LaTeX source; generates structured improvement tasks.
- *Meta-Debugger*: A self-diagnostic agent that modifies the framework itself—unlike the other agents, which modify the paper—by rewriting agent prompts, blacklisting failing strategies, or adjusting the Goal Anchor.

B. Reviewer Evaluation

We evaluate generated papers directly as complete paper artifacts: ARK targets the idea-to-paper process, so the relevant metric is manuscript quality under review criteria rather than algorithm performance on a separately scorable task. We therefore use reviewer systems that return numerical, cross-paper-comparable scores for complete paper PDFs.

Evaluated systems. We compare ARK with the two paper-generating baselines, ARIS and The AI Scientist, each using its own source code (accessed 2026-04-14). AutoResearch is excluded here because it does not generate papers by design: its agent only iterates on a training script to optimize a training metric.

Idea selection and generation configuration. ARK and ARIS receive the same research idea, SafeClaw, a security study of the OpenClaw personal AI assistant platform (Steinberger & the OpenClaw community, 2026), consisting of an empirical audit of malicious-skill prevalence on ClawHub, a community-run skill repository for OpenClaw, and a four-stage open defense pipeline: detection, automatic agent repair, least-privilege inference, and runtime policy enforcement. The AI Scientist cannot accept an external idea because its pipeline is hard-coded to three built-in templates: NanoGPT, 2D Diffusion, and Grokking. We run it on its NanoGPT template, where it produces and then runs layer-wise learning-rate adaptation, assigning each transformer layer its own learning rate instead of a single global rate. Showing each system on the kind of idea its design supports is fairer than forcing The AI Scientist into an idea space it cannot represent. Table 1 summarizes the generation configuration used before applying the reviewer systems. All three runs use a single NVIDIA A100.

Table 1. Generation setup for the evaluated paper artifacts.

System	Research idea / template	LLM configuration	Venue template
ARK	SafeClaw: empirical audit and defense pipeline for OpenClaw skills	Claude Opus 4.7	ICML 2025, double-column
ARIS	Same SafeClaw idea as ARK	Claude Opus 4.7; GPT 5.4 via OpenAI Codex for cross-model review	ICML 2024, double-column
The AI Scientist	NanoGPT built-in template; generated a layer-wise learning-rate adaptation study	Claude Opus 4.7	ICLR 2024, single-column

Reviewer setup. Table 2 specifies the three reviewer systems used in the evaluation.

Table 2. Reviewer systems used for paper-artifact evaluation.

Reviewer	Source / service	Backend model	Role
ARK reviewer	ARK's own *Reviewer* agent, sourced from `https://github.com/kaust-ark/ARK/blob/main/ark/templates/agents/reviewer.prompt`	Claude Sonnet 4.6	Self-baseline
Sakana AI Scientist reviewer (Sakana reviewer)	Reviewer implemented in `ai_scientist/perform_review.py` from the SakanaAI AI-Scientist repository (`https://github.com/SakanaAI/AI-Scientist/blob/main/ai_scientist/perform_review.py`)	Claude Sonnet 4.6	Baseline-system reviewer
CSPaper.org reviewer	Closed third-party commercial reviewer service at `https://cspaper.org`	Gemini-3-Flash	External commercial reviewer

Overall results. Figure 2 visualizes normalized overall scores, and Table 3 reports the raw overall scores and decisions. Across the three reviewer pipelines, the resulting ordering is ARK $>$ ARIS $\geq$ The AI Scientist. The ARK reviewer is included as a self-baseline for transparency; the Sakana reviewer provides a baseline-system reviewer; and CSPaper.org provides a closed third-party service that we did not author or modify.

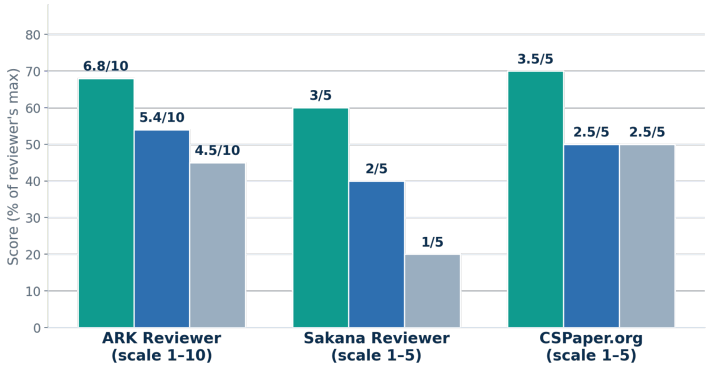

Figure 2. Normalized reviewer scores for generated papers. Scores are normalized by each reviewer's maximum scale to compare the 1–10 ARK reviewer with the 1–5 Sakana reviewer and CSPaper.org reviewer.

Table 3. Overall reviewer results across generated papers.

Paper	ARK reviewer (1–10)	Sakana reviewer (1–5)	CSPaper.org reviewer (1–5)
ARK on SafeClaw	6.8 / 10 (Weak Accept)	3 / 5 (Findings)	3.5 / 5 (Borderline Conference)
ARIS on SafeClaw	5.4 / 10 (Weak Reject)	2 / 5 (Reject)	2.5 / 5 (Borderline Findings)
The AI Scientist on NanoGPT	4.5 / 10 (Reject)	1 / 5 (Strong Reject)	2.5 / 5 (Borderline Findings)

Per-dimension scores. Tables 4 to 6 list the dimension-level scores underlying the overall summary for the Sakana reviewer, CSPaper.org reviewer, and ARK reviewer, respectively.

Calibration uncertainty. These automated reviewer scores should not be read as human peer-review ground truth. LLM-as-reviewer systems can be biased and noisy, and their score calibration is not established across venues, models, or paper domains. We treat the agreement in rank order across reviewer systems as a directional signal rather than definitive evidence of peer-review acceptance.

C. Implementation Status and Roadmap

Table 7 summarizes the implementation status of every component described in this paper. We distinguish features already running in production (✓), partially implemented or under integration (•), and roadmap items planned for upcoming releases (○).

Table 4. Sakana reviewer scores on 1–5 scales.

Dimension	**ARK**	**ARIS**	**AI Scientist**
Soundness	3	3	2
Excitement	3	2	1
Reproducibility	3	4	4
Layout / Format Compliance	4	2	3
Overall	3	2	1
Desk Reject	No	No	No
Decision	Findings	Reject	Reject

Table 5. CSPaper.org reviewer scores on 1–5 scales.

Dimension	**ARK**	**ARIS**	**AI Scientist**
Soundness	3	3	2.5
Excitement	3.5	3.0	2.5
Reproducibility	4	4	4
Datasets	3	3	3
Software	4	3	3
Overall Assessment	3.5	2.5	2.5
Desk Rejection	Pass (Borderline)	Pass (Borderline)	Pass (Borderline)
Decision	Borderline Conference	Borderline Findings	Borderline Findings

Table 6. ARK reviewer weighted scores on 1–10 scales.

Dimension	**ARK**	**ARIS**	**AI Scientist**
Technical Quality (30%)	6	5	3
Paper Presentation (30%)	8	5	6
Innovation (20%)	6	5	3
Writing Quality (20%)	7	7	7
Total (weighted)	6.8	5.4	4.5
Rating	Weak Accept	Weak Reject	Reject

D. Comparison with Existing AI Scientist Systems

We position ARK against three representative AI scientist systems, which differ in scope, design philosophy, and delivery model.

The AI Scientist (Lu et al., 2026) pioneered end-to-end autonomous paper generation and remains the reference point for fully automated pipelines. Its design prioritizes rapid, parallel idea exploration within a fixed experimental sandbox, at the cost of human steering, venue flexibility, and infrastructure scaling (e.g., the pipeline runs only on a local GPU).

AutoResearch (Karpathy, 2026) is a minimal autonomous ML training harness instead of a paper-generation system, where an external agent iterates on a single training script under a fixed time budget, optimizing a validation metric. It therefore sits largely orthogonal to the paper-generation axes of this comparison, and is included mainly for completeness.

ARIS (Auto-Research-In-Sleep) (Yang et al., 2026) is an autonomous research-automation framework designed to run unattended research pipelines end-to-end, which is the closest in scope to ARK. It covers multi-venue paper generation, literature survey, citation fetching, iterative review, persistent memory, and many LLM backends. The differences with ARK are architectural rather than feature-level: ARIS enforces quality invariants through *prompt-level discipline* rather than architectural guarantees in code, so hallucination and drift can be reduced but not structurally prevented; ARIS is delivered as a self-hosted Claude-Code-skill pipeline that users install and run themselves, whereas ARK is delivered as a managed Research-as-a-Service portal with a real-time web dashboard; and its compute targets cloud rentals (Vast.ai / Modal) and a vendor-specific HPC CLI rather than native HPC/Slurm. Specific ARK innovations such as visual-grounded review, sub-page-resolution page-length control, and content–claim alignment for citations are also absent.

ARK's contribution to this landscape is a new positioning and a new design. *Positioning*: reframing the AI scientist as a co-scientist that offloads the labor while the researcher steers the science, instead of an autonomous agent that replaces the researcher. *Design*: a model-agnostic, multi-agent pipeline in which architectural guarantees, rather than prompt-level discipline, ensure quality and truthfulness, with the full stack delivered as a managed Research-as-a-Service rather than a CLI-only prototype.

Table 8 summarizes the feature-level differences between ARK and the three existing AI scientist systems, based on direct inspection of their source code (accessed 2026-04-14).

E. Responsible Use and Disclosure

ARK is intended to assist, not replace, accountable human researchers. ARK-generated manuscripts should disclose AI assistance, and users remain responsible for checking claims, experiments, citations, and venue policies before submission. The service terms require such disclosure for generated papers.

Table 7. Implementation status of ARK. ✓ = implemented; ● = partial / under integration; ○ = planned.

Component	**Status**
System Design	
Three-phase pipeline (Initialization → Development → Review)	✓
Specialized agents (*Researcher*, *Planner*, *Experimenter*, . . .)	✓
Deep Research integration (Gemini)	✓
Memory System + Goal Anchor	✓
Pluggable LLM backend (Claude Code / Codex / Gemini)	✓
FastAPI + SQLite backend	✓
Fernet encryption of stored user API keys	✓
Cloudflare Access allowlist	✓
Token & cost transparency (per-call tokens + USD)	✓
Multi-pass LaTeX compilation with *Writer*-driven error recovery	✓
Sub-page length control via `.aux` introspection	✓
Figure generation (PaperBanana + Nano Banana)	✓
Key Technical Innovations	
Human-in-the-Loop Control	
Telegram + web portal	✓
Proactive blocker notification (waits for human input)	✓
Hallucination Validation	
API-first citations (LLM forbidden to author BibTeX)	✓
Content–claim alignment for citations (non-LLM enforced)	✓
Reverse citation verification (non-LLM enforced)	✓
Visual-grounded review (page images + source)	✓
Agent Specialization	
Project-Specific Knowledge injection per agent	✓
Retrieval-Augmented Skills	
Modular skill library with retrieval selection	✓
Skills for physical-world research (robotics / chemistry / biology)	○
World-model backend integration (embodied research)	○
Evaluation	
Self-review loop (four-dimension scoring)	✓
External reviewer evaluation	✓
Standardized benchmark suite	●
Ablation studies	○
End-to-end regression tests	○
Research-as-a-Service	
Click-to-start web portal	✓
Real-time score / token / cost dashboard	✓
Per-project sandbox: Local (Conda)	✓
Per-project sandbox: Managed Cluster (Slurm + Apptainer)	✓
Per-project sandbox: Cloud (Kubernetes + containerd)	✓
End-to-end encryption of user data in transit	○

Table 8. Feature comparison of AI scientist systems. ✓ = supported; ● = partially supported; × = not supported; N/A = not applicable (the system does not target this capability).

Feature	ARK	AI Scientist	AutoResearch	ARIS
Core AI-Scientist Capabilities				
End-to-end paper generation (LaTeX + PDF)	✓	✓	×	✓
Multi-agent architecture with role specialization	✓	●	×	✓
Grounded experiment execution on configurable backends	✓	●	●	✓
Automatic LaTeX compilation with error recovery	✓	✓	N/A	✓
Automated multi-source literature survey	✓	●	×	✓
BibTeX sourced from academic APIs	✓	●	N/A	✓
Multi-venue LaTeX templates	✓	×	N/A	✓
Venue-aware figure generation	✓	×	N/A	●
Iterative review loop with stagnation detection	✓	●	●	✓
Pluggable LLM backend	✓	●	N/A	✓
ARK-Specific Innovations				
Human-in-the-Loop Control				
Mid-run human steering interface	✓	×	●	✓
Proactive blocker notification (waits for human input)	✓	×	×	×
Hallucination Validation				
API-first citations (LLM forbidden to author BibTeX)	✓	×	N/A	●
Content–claim alignment for citations (non-LLM enforced)	✓	×	N/A	×
Reverse citation verification (non-LLM enforced)	✓	×	N/A	×
Visual-grounded review (page images and source)	✓	×	N/A	✓
Precise page-length control (`.aux` introspection)	✓	×	N/A	●
Agent Specialization				
Project-specific knowledge injection per agent	✓	×	×	×
Retrieval-Augmented Skills				
Modular skill library with retrieval selection	✓	×	×	×
Research-as-a-Service				
Fully managed web portal	✓	×	×	×
Real-time score / token / cost dashboard	✓	×	N/A	×
Per-project environment sandboxing	✓	×	×	●
HPC / Slurm integration	✓	×	×	●