Toward Autonomous Long-Horizon Engineering for ML Research

Anonymous ACL submission

Abstract

Agentic systems increasingly automate pieces of AI research. Yet turning underspecified research objectives into runnable, experimentally validated ML systems remains a central bottleneck. We study this operational setting as *long-horizon ML research engineering*: converting a research specification into a runnable ML system through repeated implementation, experimentation, and refinement. The central challenge is to sustain cumulative project progress across heterogeneous stages under delayed, confounded feedback. We introduce AiScientist, a multi-agent system built around *thin control over thick state*: a lightweight hierarchical research team coordinates through a File-as-Bus workspace that preserves decision-relevant artifacts across roles and invocations. On PaperBench, AiScientist improves over the strongest matched baselines by 9.92 and 11.15 points with Gemini-3-Flash and GLM-5, respectively. On MLE-Bench Lite, it reaches 81.82 Any Medal% under both backbones, improving over the strongest matched baselines by 4.55 and 16.67 points, and exceeding a Codex/GPT-5.5 xhigh frontier harness reference by 13.64 Any Medal points. Ablations and process analyses show that durable project state is central to later-round refinement: removing File-as-Bus lowers PaperBench score by 6.41 points and MLE-Bench Lite Any Medal% by 31.82 points. These results suggest that long-horizon AI research is not only a problem of stronger local reasoning, but a systems problem of maintaining cumulative, inspectable project progress.

1 Introduction

A central bottleneck in automating AI research is not only generating promising ideas, but turning underspecified research objectives into runnable, experimentally validated ML systems. Recent agentic systems have made rapid progress on research assistance and automation, including idea generation, literature synthesis, code generation, targeted experimentation, and scientific writing (Lu et al., 2024; Yamada et al., 2025; Tang et al., 2025; Schmidgall et al., 2025; Xu et al., 2026). Yet much of the practical difficulty of AI research lies in the engineering loop that connects these pieces: interpreting a research specification, constructing a working implementation, setting up data and environments, running experiments, diagnosing failures, and refining the system until empirical evidence supports the intended claim. We study this operational setting as *long-horizon ML research engineering*.

This setting is difficult because progress is cumulative under delayed feedback. A paper or research specification rarely determines all implementation choices. Important details may be implicit, missing, or scattered across sections, datasets and pretrained models must be located and integrated, and experimental discrepancies often appear only after hours of implementation and execution. When a run fails or a metric does not match the target, the cause may lie in paper interpretation, data processing, model design, training configuration, infrastructure, or some interaction among them. Thus the challenge is not simply to solve a sequence of local engineering subproblems, but to preserve enough evolving project state for later decisions to remain coherent. This difficulty is visible in rigorous evaluation: on PaperBench, the best reported agent achieves only 21% of the replication rubric, compared with 41% achieved by top ML PhDs under a 48-hour budget (Starace et al., 2025).

The core systems problem is that agent invocations are transient, while research progress must be cumulative and persistent. This creates two continuity requirements. First, *role continuity*: when a specialist is invoked again later, it must be able to resume from prior role-level work, including what earlier invocations tried, which assumptions they made, which failures they observed, and what questions they left open. Second, *project continuity*: different roles must coordinate around shared

evidence produced across the whole project, rather than around lossy conversational handoffs. Existing agent organizations struggle with these requirements in complementary ways. Single-agent systems must carry an expanding history of analysis, code changes, logs, and diagnostic reasoning in a limited active context. Multi-agent systems can distribute work, but often transfer state through compressed handoffs that may omit the detailed evidence needed for later debugging and refinement (Cemri et al., 2025; Yan et al., 2025). For long-horizon ML research engineering, the key question is therefore how to convert *transient agent activity* into *durable, inspectable project progress*.

We introduce **AiScientist**, a system for autonomous long-horizon engineering for ML research designed to satisfy these two continuity requirements through *thin control over thick state*. The top-level Orchestrator provides thin control: it manages stage-level progress through concise summaries, high-level directives, and a compact workspace map, without carrying the full project history in its active context. The shared workspace provides thick state through a **File-as-Bus protocol**: decision-relevant state is externalized into persistent project artifacts rather than left in transient conversations. This artifact substrate supports *role continuity* because a newly invoked specialist can inspect its role-owned artifacts, such as implementation rationales or experimental notes, to resume prior work. It supports *project continuity* because specialists ground their decisions in the same system of record, including paper analyses, plans, experiment records, failure diagnoses, and result summaries. A **hierarchical research team** ties these pieces together: the Orchestrator delegates to specialized agents for paper comprehension, task prioritization, implementation, and experimentation, while durable artifacts carry detailed state across invocations and roles.

We evaluate whether this design yields sustained empirical progress on two complementary benchmarks: PaperBench (Starace et al., 2025) for from-scratch paper replication and MLE-Bench Lite (Chan et al., 2025) for iterative competition-style ML improvement. Across matched comparisons, AiScientist improves over the strongest baselines by 9.92/11.15 points on PaperBench with Gemini-3-Flash/GLM-5, and reaches 81.82 Any Medal% on MLE-Bench Lite under both backbones, improving by 4.55/16.67 points. To calibrate these results against a frontier harness, we also report Codex (OpenAI, 2025) with GPT-5.5 xhigh as a reference. The best AiScientist exceeds it by 4.28 points on PaperBench and 13.64 Any Medal points on MLE-Bench Lite. Ablations support the continuity claim: removing File-as-Bus lowers PaperBench score by 6.41 points and MLE-Bench Lite Any Medal% by 31.82 points, with the largest degradation appearing in later-round refinement rather than first-pass executability.

To summarize, our contributions are as follows:

- We formulate long-horizon ML research engineering as a cumulative project-state problem, identifying *role continuity* and *project continuity* as central requirements for sustained progress under delayed experimental feedback.
- We present AiScientist, a multi-agent system built around *thin control over thick state*, where lightweight orchestration is paired with a File-as-Bus protocol that externalizes decision-relevant state into durable project artifacts.
- We provide empirical evidence on PaperBench and MLE-Bench Lite, including matched baselines, a frontier harness reference, and ablations showing that durable state continuity is a key bottleneck for long-horizon research engineering.

2 Task Formulation

We formulate *long-horizon ML research engineering* as the task of turning a research specification into a runnable ML system. Given a research specification, an environment, a resource-access policy, and a time budget, the agent must produce a submission. Evaluation is performed by fresh execution in a clean environment, measuring both executability and whether its empirical behavior satisfies the target objective. This task is challenging along four dimensions:

- **Underspecification:** The research specification is typically underspecified rather than a complete blueprint. Implementation details may be implicit, scattered across sections, or omitted entirely, so the agent must recover missing decisions from incomplete specifications, related literature, and other permitted public resources.
- **System Setup Burden:** Success depends on substantial system setup beyond code alone, including configuring environments, acquiring datasets and models from permitted sources, and integrating these resources into a runnable system.
- **Delayed Feedback:** Meaningful evidence ar-

rives only after experiments run, and discrepancies may stem from interpretation, implementation, data processing, or infrastructure. The agent must reason from delayed and often confounded feedback before deciding what to fix next.

- **State Continuity:** Each round of implementation and experimentation produces code, configurations, logs, results, and diagnostic evidence that later decisions must correctly interpret and build on. Progress depends on maintaining continuity across heterogeneous stages over long horizons.

3 AiScientist

3.1 Overview: Thin Control over Thick State

AiScientist is built to sustain progress when agent invocations are transient but research progress must remain cumulative. It operationalizes *thin control over thick state* through two coupled mechanisms: a File-as-Bus protocol for durable state continuity (§3.2) and a hierarchical research team for lightweight stage-level control (§3.3). As shown in Figure 1, these mechanisms target both *role continuity*, where later invocations of a specialist resume prior role-level work, and *project continuity*, where different specialists coordinate around shared evidence rather than lossy conversational handoffs.

The coupling is important. File-as-Bus makes detailed project state durable and inspectable, but it does not by itself decide which evidence matters next or which role should act. The hierarchical research team provides that control, but keeps it thin by routing decisions through compact summaries and the workspace map rather than the full project history. Together, the two mechanisms let AiScientist preserve detailed state without forcing every control decision to carry it.

3.2 File-as-Bus Coordination

AiScientist implements thick state through a File-as-Bus protocol. File-as-Bus is not merely shared storage, but an artifact-mediated coordination protocol. Agents coordinate by publishing and consuming durable artifacts, so that project evidence, decisions, and execution state remain inspectable across invocations. The workspace therefore acts as the system of record for project progress.

Let σ denote the File-as-Bus schema, which specifies each artifact's purpose, update mode, permitted writers and readers. Let W_t denote the durable workspace state at step t. The workspace map is a compact runtime view of both the current artifact state and the schema:

$$m_t = \mathcal{M}(W_t; \sigma), \tag{1}$$

where $\mathcal{M}$ combines the current artifact state in W_t with schema metadata from σ to produce a compact index of available artifacts and their roles. Agents start from m_t to identify relevant artifacts, then inspect task-specific parts of W_t on demand.

The schema σ distinguishes three update modes to match different lifecycles: *append-only logs* preserve chronological evidence such as implementation logs and experiment diagnoses; *versioned artifacts* maintain a canonical current state while retaining prior revisions, such as plans, analyses, and reports; and *mutable state* stores short-lived control information that does not require history. This distinction matters for long-horizon work: some evidence should never be overwritten, some artifacts need a stable current version with recoverable history, and some state should remain lightweight.

Together, the schema and workspace map support *role continuity* by giving each specialist persistent access to role-relevant artifacts: later invocations can inspect prior notes, unresolved blockers, and role-specific logs before acting. They also support *project continuity* by making cross-role dependencies explicit: paper comprehension informs prioritization, prioritization constrains implementation, implementation produces executable artifacts, and experimentation writes back evidence for later refinement. In this sense, File-as-Bus is both a durable state substrate and a coordination channel.

3.3 Hierarchical Research Team

File-as-Bus determines how project state is preserved and exposed, and the hierarchical research team determines how work is routed. At the top level, the Orchestrator maintains a compact control context, monitors stage-level progress, and decides which bottleneck to address next. Its outputs are concise directives to specialists rather than detailed procedural scripts, leaving each specialist to expand into the local context needed for its stage.

The key control abstraction is *Agent-as-Tool*. The Orchestrator treats specialist invocation as part of the same action space as ordinary tool use:

$$a_t = \pi_0(c_t, m_t), \qquad a_t \in \mathcal{T}_0 \cup \mathcal{A}_1, \tag{2}$$

where c_t is the Orchestrator's control context, $\mathcal{T}_0$ is its native tool set, and $\mathcal{A}_1$ is the set of Tier-1 specialists. This makes delegation selective: the

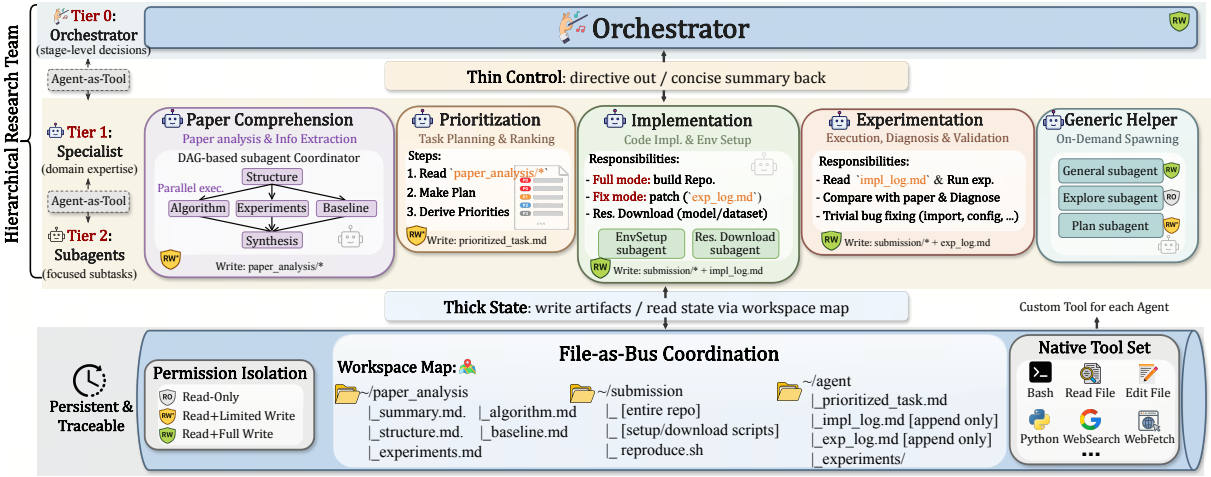


Figure 1: Overview of AiScientist. A top-level Orchestrator maintains thin control through stage-level directives, concise summaries, and a compact workspace map. Specialized agents perform paper comprehension, task prioritization, implementation, experimentation, and auxiliary exploration, while coordinating through a File-as-Bus workspace that serves as the system of record. This design turns transient agent invocations into durable role-level and project-level progress by externalizing decision-relevant state into persistent project artifacts.

Orchestrator can perform lightweight operations directly and invoke a specialist when the next step requires focused expertise or a longer local horizon.

When specialist π_j is invoked with directive d_t, it receives the directive and workspace map, reads task-relevant artifacts from W_t, and writes back both a concise summary and workspace updates:

$$(s_t, \Delta W_t) = \pi_j(d_t, m_t; W_t), \tag{3}$$

$$W_{t+1} = \mathcal{U}(W_t, \Delta W_t). \tag{4}$$

Here, $\mathcal{U}$ applies the specialist's artifact updates to the File-as-Bus workspace. The summary s_t updates the Orchestrator's compact control view, while ΔW_t preserves detailed progress in the workspace. This separation lets specialist work remain rich locally without forcing the Orchestrator to carry the full reasoning trace.

The Tier-1 specialists align with the major stages of ML research engineering:

- **Paper Comprehension Specialist**: extracts implementation-relevant details, target metrics, proposed methods, baselines, ambiguities, and assumptions from the research specification.
- **Prioritization Specialist**: converts paper understanding into an ordered execution plan, ranking tasks by dependency, impact, and feasibility.
- **Implementation Specialist**: builds or patches the system, handles setup and resource integration, and records major code-side decisions.
- **Experimentation Specialist**: executes the pipeline, compares outcomes against target objectives, and records results, failures, and diagnoses.
- **Generic Helper Interface**: supports focused auxiliary work such as resource lookup, exploration, or local planning.

Specialists may use tightly scoped Tier-2 subagents for bounded subtasks. The hierarchy is not recursive: Tier-2 outputs are folded back into the invoking specialist's local context and then written to durable artifacts when they matter for later stages.

3.4 Evidence-Driven Research-Engineering Loop

AiScientist uses the mechanisms above to run an adaptive research-engineering loop rather than a rigid one-pass pipeline. Early stages turn an underspecified research objective into an execution plan and a runnable scaffold: code, configuration, setup path, resource acquisition process, and an entry point that can be repeatedly extended and evaluated. Once this scaffold exists, progress is driven by alternating implementation and experimentation, with each run producing executable evidence such as failures, partial successes, metric gaps, bottlenecks, and result discrepancies. The Orchestrator uses the workspace map and recorded evidence to choose the next intervention, such as patching implementation, revising data processing or configuration, rerunning experiments, or returning to earlier analysis when an assumption is invalidated.

In this way, delayed feedback becomes cumulative progress. Once failures are recorded as durable artifacts, they become inspectable project evidence rather than isolated local errors. Suc-

Task Name	GPT-5.5	Gemini-3-Flash				GLM-5			
	Codex	BasicAgent	IterAgent	AiScientist	Δ	BasicAgent	IterAgent	AiScientist	Δ
adaptive-pruning	33.42	24.53	3.05	**27.25**	+2.72	30.82	11.93	**33.26**	+2.44
all-in-one	52.93	20.86	45.13	**46.29**	+1.16	33.78	44.43	**49.47**	+5.04
bam	56.65	48.46	45.04	**56.59**	+8.13	51.45	47.91	**61.11**	+9.66
bbox	17.30	15.43	8.30	**33.79**	+18.36	23.55	19.28	**30.02**	+6.47
bridging-data-gaps	14.16	12.59	12.44	**23.09**	+10.50	9.80	12.50	**26.46**	+13.96
fre	14.06	21.67	23.89	**35.21**	+11.32	21.60	16.67	**28.98**	+7.38
ftrl	1.50	5.87	4.15	**10.11**	+4.24	3.71	6.70	**8.34**	+1.64
lbcs	26.60	17.75	15.26	**27.90**	+10.15	20.68	22.74	**30.10**	+7.36
lca-on-the-line	22.08	12.97	18.30	**30.23**	+11.93	22.55	26.15	**28.53**	+2.38
mechanistic-understanding	45.67	14.86	21.89	**29.95**	+8.06	32.49	34.96	**40.55**	+5.59
pinn	40.64	26.63	30.81	**49.92**	+19.11	22.18	25.77	**58.76**	+32.99
rice	7.94	10.43	8.88	**10.87**	+0.44	6.56	0.27	**10.18**	+3.62
robust-clip	30.14	15.45	10.43	**18.28**	+2.83	22.43	27.56	**28.66**	+1.10
sample-specific-masks	57.11	25.39	33.34	**36.77**	+3.43	36.93	41.26	**44.13**	+2.87
sapg	18.15	11.45	12.65	**19.85**	+7.20	6.99	4.95	**31.69**	+24.70
sequential-neural	41.67	53.51	60.24	**64.94**	+4.70	27.2	35.53	**49.32**	+13.79
stay-on-topic	32.31	8.37	13.69	**20.13**	+6.44	3.69	8.81	**14.81**	+6.00
stochastic-interpolants	41.12	17.04	17.37	**18.81**	+1.44	32.18	28.06	**42.10**	+9.92
test-time-model-adaptation	26.28	15.27	18.13	**32.45**	+14.32	17.81	21.19	**27.33**	+6.14
what-will-my-model-forget	9.35	6.61	8.99	**17.87**	+8.88	25.14	10.75	**30.82**	+5.68
Average Score	29.45	19.26	20.60	**30.52**	+9.92	22.58	22.37	**33.73**	+11.15
Avg Cost / Task	$33.05	$6.25	$27.44	$15.67	-	$4.90	$54.90	$12.20	-

Table 1: Main results on PaperBench full evaluation. Codex with GPT-5.5 xhigh is shown as a frontier harness reference; red values indicate AiScientist's gains over the strongest matched baseline. **Bold** and underlined denote the best and second-best results within each backbone block.

cessful changes are likewise preserved as runnable code, setup state, logs, and result records. Later invocations can continue from prior evidence instead of rediscovering it, enabling long-horizon refinement under finite time budgets.

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluate AiScientist in two complementary long-horizon ML research engineering benchmarks. **PaperBench** (Starace et al., 2025) evaluates from-scratch replication of top-tier ML papers. **MLE-Bench Lite** (Chan et al., 2025) evaluates sustained improvement on competition-style ML tasks, with *Any Medal%* as the primary metric. Together, these benchmarks test whether an agent can maintain coherent progress across heterogeneous stages under realistic time budgets, rather than succeed only in a single narrow setting.

Baselines. On PaperBench, we compare against BasicAgent and IterativeAgent (Starace et al., 2025) under the same evaluation protocol. On MLE-Bench Lite, we report controlled comparisons against strong autonomous ML engineering systems with diverse designs, including AIDE (Jiang et al., 2025), LoongFlow (Wan et al., 2025), and ML-Master 2.0 (Zhu et al., 2026). We also include two contextual reference sets that are not treated as matched baselines: official MLE-Bench Lite leaderboard results (Yang et al., 2025; Li et al., 2025a; Liu et al., 2025; Toledo et al., 2025; Nadafian et al., 2026; Chen et al., 2026b; Zhang et al., 2026), and a frontier Codex/GPT-5.5 xhigh harness (OpenAI, 2025) evaluated under our setup.

Implementation Details. We instantiate AiScientist with two backbone LLMs, Gemini-3-Flash (Google DeepMind, 2025) and GLM-5 (Zeng et al., 2026). Across both benchmarks, each run is allocated one H20 GPU and a 24-hour budget per task, matching the standard setting. For PaperBench full evaluation, we follow the official evaluation protocol (Starace et al., 2025) and use GPT-5.4 (OpenAI, 2026) as the grading model. Under this grading setup, a full 20-task PaperBench evaluation costs approximately $832, which materially limits large-scale repeated evaluation. For MLE-Bench Lite, we follow the MLE-Bench convention and report mean ± SEM over 3 runs/seeds.

4.2 Main Results on PaperBench

Table 1 reports the full PaperBench evaluation. Across both backbones, AiScientist consistently improves over the strongest matched baseline: by 9.92 points with Gemini-3-Flash and by 11.15 points with GLM-5. These gains are obtained at substantially lower cost than IterativeAgent: $15.67 versus $27.44 per task under Gemini-3-Flash, and $12.20 versus $54.90 under GLM-5. The comparison is informative because IterativeAgent already increases interaction relative to BasicAgent, yet remains well below AiScientist while spending more.

Agent	Model	Valid Submission	Above Median	Bronze	Silver	Gold	Any Medal
Official MLE-Bench Leaderboard Results							
InternAgent	Deepseek-R1	100.00 ±0.00	78.79 ±5.46	10.61 ±1.52	16.67 ±3.03	34.85 ±1.52	62.12 ±3.03
ML-Master	Deepseek-R1	100.00 ±0.00	74.24 ±1.52	4.55 ±2.62	13.64 ±0.00	30.30 ±3.03	48.48 ±1.52
AIRA-dojo	o3	100.00 ±0.00	70.45 ±1.60	7.95 ±0.86	12.73 ±1.42	34.32 ±1.02	55.00 ±1.47
ML-Master 2.0	Deepseek-V3.2-Spe	100.00 ±0.00	84.85 ±1.52	13.64 ±2.62	31.82 ±5.25	30.30 ±3.03	75.76 ±1.52
R&D-Agent	GPT-5	77.27 ±0.00	74.24 ±1.52	12.12 ±4.01	22.73 ±0.00	33.33 ±3.03	68.18 ±2.62
Famou-Agent 2.0	Gemini-2.5-Pro	100.00 ±0.00	86.36 ±2.62	15.15 ±4.01	19.70 ±4.01	40.91 ±2.62	75.76 ±1.52
MARS	Gemini-3-Pro	100.00 ±0.00	89.39 ±1.52	6.06 ±1.52	15.15 ±1.52	53.03 ±1.52	74.24 ±1.52
Leeroo	Gemini-3-Pro	68.18 ±2.62	68.18 ±2.62	18.18 ±2.62	19.70 ±4.01	30.30 ±1.52	68.18 ±2.62
AIBuildAI	Claude-Opus-4.6	100.00 ±0.00	81.82 ±0.00	13.64 ±6.94	25.76 ±4.01	37.88 ±4.01	<u>77.27</u> ±0.00
Frontier Harness							
Codex	GPT-5.5 (xhigh)	100.00 ±0.00	81.82 ±0.00	1.52 ±1.52	19.70 ±1.52	46.97 ±4.01	68.18 ±2.62
Controlled Evaluation							
AIDE	Gemini-3-Flash	87.88 ±5.46	59.09 ±2.62	7.58 ±1.52	9.09 ±0.00	28.79 ±1.52	45.45 ±0.00
LoongFlow	Gemini-3-Flash	77.27 ±0.00	77.27 ±0.00	12.12 ±5.46	25.76 ±5.46	39.39 ±3.03	<u>77.27</u> ±0.00
AiScientist (Ours)	Gemini-3-Flash	100.00 ±0.00	86.36 ±0.00	16.67 ±1.52	25.76 ±3.03	39.39 ±4.01	**81.82** ±0.00
AIDE	GLM-5	66.67 ±5.46	42.42 ±4.01	7.58 ±3.03	6.06 ±4.01	21.21 ±1.52	34.85 ±3.03
ML-Master 2.0	GLM-5	100.00 ±0.00	80.30 ±1.52	16.67 ±1.52	16.67 ±1.52	31.82 ±0.00	65.15 ±1.52
AiScientist (Ours)	GLM-5	100.00 ±0.00	89.39 ±1.52	13.64 ±2.62	25.76 ±3.03	42.42 ±1.52	**81.82** ±0.00

Table 2: Main results on MLE-Bench Lite. Values are percentages reported as mean ± SEM over three runs/seeds following the MLE-Bench convention. Official leaderboard rows are contextual references, the Codex/GPT-5.5 xhigh row is a frontier harness reference evaluated under our setup, and controlled evaluation rows are matched comparisons. **Bold** and <u>underlined</u> denote the best and second-best *Any Medal* performance.

This suggests that long-horizon performance is not explained by more rounds alone. Those rounds must preserve and reuse prior project evidence. As an additional contextual reference, the best AiScientist backbone exceeds the Codex harness by 4.28 points on PaperBench average score.

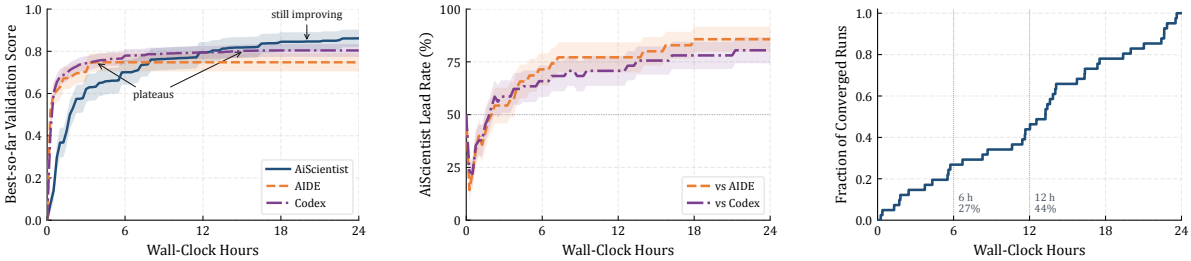
4.3 Main Results on MLE-Bench Lite

Table 2 reports MLE-Bench Lite results. In controlled comparisons, AiScientist reaches 81.82 Any Medal% under both backbones, improving over the strongest matched baseline by 4.55 points with Gemini-3-Flash and 16.67 points with GLM-5. The gains are also reflected in Above Median%, where AiScientist improves by 9.09 points under both backbones. These results indicate that AiScientist not only produces valid submissions, but also sustains the iterative improvement needed to obtain competitive outcomes. Relative to contextual references, AiScientist exceeds the Codex/GPT-5.5 xhigh harness by 13.64 Any Medal points and surpasses the strongest official leaderboard Any Medal result reported in Table 2.

4.4 Long-Horizon Improvement Dynamics

Final metrics alone obscure when progress occurs. Figure 2 separates three aspects of long-horizon improvement on MLE-Bench Lite. The average score curve measures the magnitude of performance over time; the pairwise lead-rate curve measures how broadly AiScientist is ahead across matched task–seed comparisons; and the convergence-time curve measures when runs actually reach their final best score. AIDE and Codex improve rapidly early in the run, with Codex especially strong in the first few hours. By contrast, AiScientist improves more slowly at the beginning, consistent with spending early budget on task understanding, planning, and scaffold construction, but continues improving after both references largely plateau.

The pairwise and convergence views clarify why this is a long-horizon effect rather than a late gain on a few outlier tasks. AiScientist can lead on a majority of paired comparisons before its mean score overtakes the references, because early wins may be narrow while some remaining losses are still large. At the same time, the time-to-final-best distribution shows that many runs do not reach their final best score early. Together, the panels show that AiScientist's advantage gradually broadens across tasks and strengthens over the full budget.



(a) Mean best-so-far score (b) Pairwise lead rate (c) Time to final best

Figure 2: Long-horizon improvement dynamics on MLE-Bench Lite under GLM-5. (a) Mean validation best-so-far normalized score with ± 1 SEM across task–seed trajectories. (b) Pairwise lead rate of AiScientist against AIDE and Codex, with binomial standard error. (c) Cumulative distribution of when runs first reach their final best score. AiScientist starts slower but continues improving after both references largely plateau. Its advantage first becomes broad across paired runs, and many runs require late-budget refinement to reach their final best.

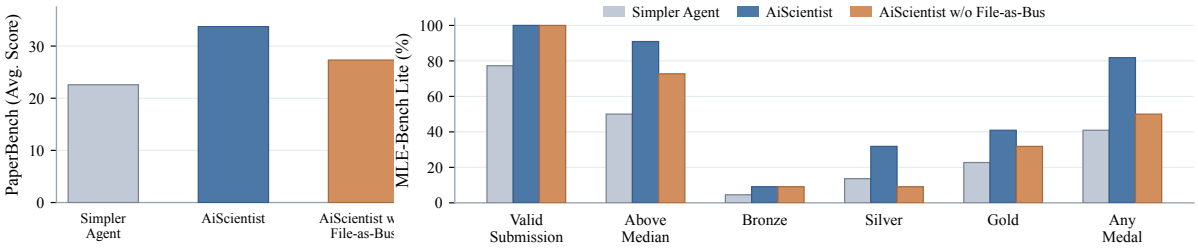


Figure 3: Mechanism analysis of AiScientist under GLM-5. Left: PaperBench ablations show that both hierarchical organization and File-as-Bus contribute to replication performance; removing File-as-Bus alone lowers score by 6.41 points. Right: on MLE-Bench Lite, File-as-Bus removal preserves first-pass validity but substantially reduces medal-level outcomes, indicating that durable state continuity primarily supports later-round refinement.

4.5 Mechanism Analysis

Figure 3 analyzes two mechanisms behind the gains of AiScientist: durable artifact-based continuity and hierarchical agent organization. Removing File-as-Bus causes large degradation on both benchmarks. On PaperBench, average score drops by 6.41 points. On MLE-Bench Lite, Any Medal drops by 31.82 points. The MLE-Bench Lite pattern is especially diagnostic: Valid Submission and Bronze remain largely intact, while Above Median, Silver, Gold, and Any Medal degrade much more. This suggests that File-as-Bus is less important for producing a minimally runnable starting point than for preserving evidence across later rounds of diagnosis and improvement.

Hierarchical organization also appears to matter. Relative to simpler non-hierarchical baselines, the advantage remains substantial even when File-as-Bus is removed: on PaperBench, AiScientist without File-as-Bus still improves average score by 4.74 points over the simpler agent baseline, while on MLE-Bench Lite it improves Above Median by 22.73 points and Any Medal by 9.09 points. Together with the IterativeAgent comparison, this indicates that the gains of AiScientist are not reducible to interaction count alone. Long-horizon ML research engineering benefits from both durable project state and a control structure that routes heterogeneous work to specialized roles.

4.6 Behavioral Analysis on PaperBench

We next analyze how different agents spend their trajectories on PaperBench. Figure 4 groups steps by workflow stage rather than by agent-specific roles, making the comparison meaningful across different methods. Codex provides an important reference point: despite using far fewer steps, it remains highly competitive, reflecting the strength of both GPT-5.5 xhigh and the Codex harness. Thus, the figure should not be read as showing that longer trajectories are intrinsically better. The key question is where those steps are spent.

Compared with IterativeAgent, AiScientist shifts substantially more of its trajectory into implementation, experimentation, and validation, while IterativeAgent's many steps are less concentrated in implementation-heavy progress. This comparison reinforces the main result: additional interaction helps only when it is organized around cumulative research-engineering work, rather than simply extending a generic agent loop.

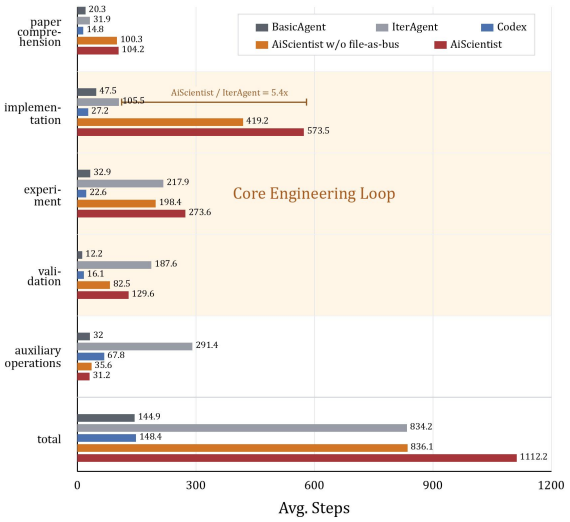


Figure 4: Stage-level workflow-step allocation on PaperBench under GLM-5. Grouping steps by workflow stage enables comparison across different methods. Codex is a strong low-step reference, while AiScientist concentrates more steps in the core engineering loop.

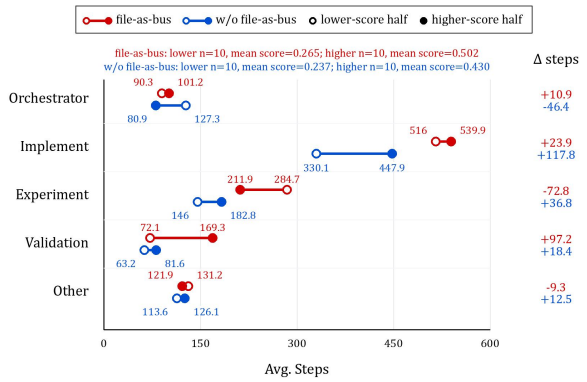


Figure 5: Workflow-step allocation for higher- and lower-scoring PaperBench trajectories under GLM-5. Tasks are split by final PaperBench score, separately for AiScientist and its variant without File-as-Bus. With File-as-Bus, higher-scoring trajectories spend more of their later effort on validation and closure; without File-as-Bus, higher-scoring trajectories use substantially more implementation and experimentation steps, suggesting less efficient reuse of prior evidence.

Figure 5 further examines the working patterns of higher- and lower-scoring trajectories. For AiScientist, higher-scoring trajectories are not characterized by more implementation or experimentation. Instead, they spend more effort on validation, while using fewer experiment steps than lower-scoring trajectories. This suggests that successful trajectories are not simply those that try more variants. Rather, they appear to reach useful implementations earlier and spend more of the later trajectory on checking, validating, and closing the loop.

The variant without File-as-Bus shows a different pattern. Without durable project state, higher-scoring trajectories require substantially more implementation and experimentation steps than lower-scoring trajectories. This is consistent with the mechanism analysis in Figure 3: when intermediate evidence is not preserved as effectively, success depends more heavily on repeated low-level execution effort. With File-as-Bus, later invocations can reuse recorded assumptions, failures, experiment results, and unresolved issues, shifting successful trajectories toward evidence reuse and validation. Together, these two behavior analyses show that AiScientist's gains are not merely a consequence of longer trajectories, but of directing long-horizon work toward cumulative project progress.

5 Related Work

Recent systems have advanced AI research across scientific discovery (Schmidgall et al., 2025; Tang et al., 2025), objective-driven ML engineering (Jiang et al., 2025; Zhu et al., 2026), and paper-to-code reproduction (Zhou et al., 2025; Seo et al., 2026). These systems show that agents can contribute to different parts of research. We build on this progress and study a complementary question: how to sustain coherent research-engineering progress over long horizons. AiScientist addresses this by combining hierarchical orchestration with artifact-mediated project state, enabling specialists to coordinate through durable evidence rather than transient conversational handoffs.

6 Conclusion

We study long-horizon ML research engineering, where agents must turn underspecified research objectives into runnable systems and sustain progress under delayed, confounded experimental feedback. AiScientist addresses this setting through thin control over thick state: hierarchical orchestration over a File-as-Bus workspace that preserves decision-relevant artifacts across roles and invocations. Across PaperBench and MLE-Bench Lite, AiScientist improves over strong matched baselines, exceeds a frontier Codex/GPT-5.5 harness reference on key metrics, and shows large degradation when durable project state is removed. Together, the results suggest that long-horizon AI research automation is not only a problem of stronger local reasoning, but a systems problem of maintaining cumulative, inspectable project progress.

Limitations

Our study intentionally focuses on the operational engineering core of AI research automation, as instantiated by PaperBench and MLE-Bench Lite, rather than attempting to cover the entire scientific workflow. These benchmarks are well matched to our focus because they stress underspecified specifications, system setup, delayed experimental feedback, and iterative refinement. They leave other important settings, such as open-ended idea generation, theoretical analysis, human collaboration, and complete paper writing, to future work.

Ethical Considerations

This work studies autonomous agents for ML research engineering. Such systems may reduce engineering burden and improve reproducibility, but agent-generated code, experiment logs, and empirical claims can still be incorrect and should be verified by fresh execution and human review before being used in scientific reporting or downstream applications. Autonomous agents can also consume substantial compute and interact with external resources; in our experiments, agents operate under benchmark-defined resource policies, fixed time budgets, and controlled execution environments. For real-world use, systems such as AiScientist should be deployed with human oversight, resource limits, and license-aware handling of external datasets, models, and code.

References

Analemma. 2026. Introducing FARS: A fully automated research system. https://analemma.ai/blog/introducing-fars/. Deployed February 13–March 3, 2026; produced 100 papers in 228 hours.

Mert Cemri, Michael Zhan Pan, Shuyang Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. 2025. Why do multi-agent LLM systems fail?

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Aleksander Madry, and Lilian Weng. 2025. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *The Thirteenth International Conference on Learning Representations*.

Guoxin Chen, Minpeng Liao, Peiying Yu, Dingmin Wang, Zile Qiao, Chao Yang, Xin Zhao, and Kai Fan. 2025. C-3PO: Compact plug-and-play proxy optimization to achieve human-like retrieval-augmented generation. In *Forty-second International Conference on Machine Learning*.

Guoxin Chen, Fanzhe Meng, Jiale Zhao, Minghao Li, Daixuan Cheng, Huatong Song, Jie Chen, Yuzhi Lin, Hui Chen, Xin Zhao, and 1 others. 2026a. Beyondswe: Can current code agent survive beyond single-repo bug fixing? *arXiv preprint arXiv:2603.03194*.

Jiefeng Chen, Bhavana Dalvi Mishra, Jaehyun Nam, Rui Meng, Tomas Pfister, and Jinsung Yoon. 2026b. Mars: Modular agent with reflective search for automated ai research. *arXiv preprint arXiv:2602.02660*.

Dayuan Fu, Shenyu Wu, Yunze Wu, Zerui Peng, Yaxing Huang, Jie Sun, Ji Zeng, Mohan Jiang, Lin Zhang, Yukun Li, and 1 others. 2026. davinci-env: Open swe environment synthesis at scale. *arXiv preprint arXiv:2603.13023*.

Google DeepMind. 2025. Gemini 3 flash. https://deepmind.google/models/gemini/flash/.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, and 1 others. 2023. Metagpt: Meta programming for a multi-agent collaborative framework. In *The twelfth international conference on learning representations*.

Peter Jansen, Oyvind Tafjord, Marissa Radensky, Pao Siangliulue, Tom Hope, Bhavana Dalvi Mishra, Bodhisattwa Prasad Majumder, Daniel S Weld, and Peter Clark. 2025. CodeScientist: End-to-end semi-automated scientific discovery with code-based experimentation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 13370–13467, Vienna, Austria. Association for Computational Linguistics.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. 2025. AIDE: AI-driven exploration in the space of code. *arXiv preprint arXiv:2502.13138*.

Andrej Karpathy. 2026. autoresearch: AI agents running research on single-GPU nanochat training automatically. https://github.com/karpathy/autoresearch.

Annan Li, Chufan Wu, Zengle Ge, Yee Hin Chong, Zhinan Hou, Lizhe Cao, Cheng Ju, Jianmin Wu, Huaiming Li, Haobo Zhang, and 1 others. 2025a. The fm agent. *arXiv preprint arXiv:2510.26144*.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008.

Zongwei Li, Zhonghang Li, Zirui Guo, Xubin Ren, and Chao Huang. 2025b. DeepCode: Open agentic coding. *arXiv preprint arXiv:2512.07921*.

Jiaqi Liu, Peng Xia, Siwei Han, Shi Qiu, Letian Zhang, Guiming Chen, Haoqin Tu, Xinyu Yang, Jiawei Zhou, Hongtu Zhu, Yun Li, Yuyin Zhou, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. 2026. AutoResearchClaw: Fully autonomous research from idea to paper. https://github.com/aiming-lab/AutoResearchClaw.

Zexi Liu, Yuzhu Cai, Xinyu Zhu, Yujie Zheng, Runkun Chen, Ying Wen, Yanfeng Wang, Siheng Chen, and 1 others. 2025. Ml-master: Towards ai-for-ai via integration of exploration and reasoning. *arXiv preprint arXiv:2506.16499*.

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

Alireza Nadafian, Alireza Mohammadshahi, and Majid Yazdani. 2026. Kapso: A knowledge-grounded framework for autonomous program synthesis and optimization. *arXiv preprint arXiv:2601.21526*.

Jaehyun Nam, Jinsung Yoon, Jiefeng Chen, Jinwoo Shin, Sercan O Arik, and Tomas Pfister. 2025. MLE-STAR: Machine learning engineering agent via search and targeted refinement. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

OpenAI. 2025. Introducing upgrades to codex. https://openai.com/index/introducing-upgrades-to-codex/.

OpenAI. 2026. Introducing GPT-5.4.

Yingming Pu, Tao Lin, and Hongyu Chen. 2025. PiFlow: Principle-aware scientific discovery with multi-agent collaboration. *arXiv preprint arXiv:2505.15047*.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186, Bangkok, Thailand. Association for Computational Linguistics.

Samuel Schmidgall and Michael Moor. 2025. AgentRxiv: Towards collaborative autonomous research. *arXiv preprint arXiv:2503.18102*.

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

Minju Seo, Jinheon Baek, Seongyun Lee, and Sung Ju Hwang. 2026. Paper2Code: Automating code generation from scientific papers in machine learning.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. 2025. Paperbench: Evaluating AI's ability to replicate AI research. In *Forty-second International Conference on Machine Learning*.

Jiabin Tang, Lianghao Xia, Zhonghang Li, and Chao Huang. 2025. AI-Researcher: Autonomous scientific innovation.

Xinyu Tang, Han Peng, Guoxin Chen, Yuze Shi, Zitao Su, Peiyu Liu, Wayne Xin Zhao, Yawen Li, and Zhe Xue. 2026. Agent systems with harness engineering.

Edan Toledo, Karen Hambardzumyan, Martin Josifoski, RISHI HAZRA, Nicolas Baldwin, Alexis Audran-Reiss, Michael Kuchnik, Despoina Magka, Minqi Jiang, Alisia Maria Lupidi, Andrei Lupu, Roberta Raileanu, Tatiana Shavrina, Kelvin Niu, Jean-Christophe Gagnon-Audet, Michael Shvartsman, Shagun Sodhani, Alexander H Miller, Abhishek Charnalia, and 6 others. 2025. AI research agents for machine learning: Search, exploration, and generalization in MLE-bench. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Chunhui Wan, Xunan Dai, Zhuo Wang, Minglei Li, Yanpeng Wang, Yinan Mao, Yu Lan, and Zhiwen Xiao. 2025. Loongflow: Directed evolutionary search via a cognitive plan-execute-summarize paradigm. *arXiv preprint arXiv:2512.24077*.

Yixuan Weng, Minjun Zhu, Qiujie Xie, QiYao Sun, Zhen Lin, Sifan Liu, and Yue Zhang. 2026. Deepscientist: Advancing frontier-pushing scientific findings progressively. In *The Fourteenth International Conference on Learning Representations*.

Tengyue Xu, Zhuoyang Qian, Gaoge Liu, Li Ling, Zhentao Zhang, Biao Wu, Shuo Zhang, Ke Lu, Wei Shi, Ziqi Wang, and 1 others. 2026. Idea2story: An automated pipeline for transforming research concepts into complete scientific narratives. *arXiv preprint arXiv:2601.20833*.

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

Bingyu Yan, Zijie Zhou, Lanqing Zhang, Liang Zhang, Zhenjie Zhou, Duanyu Miao, Zhiguo Li, Chunyuan Li, and Xiang Zhang. 2025. Beyond self-talk: A

communication-centric survey of LLM-based multi-agent systems. *arXiv preprint arXiv:2502.14321*.

Xu Yang, Xiao Yang, Shikai Fang, Yuante Zhang, Bowen Li, Jian Wang, Bofeng Xian, Qiang Li, Jian Li, and 1 others. 2025. R&D-Agent: An LLM-agent framework towards autonomous data science. *arXiv preprint arXiv:2505.14738*.

Peiying Yu, Guoxin Chen, and Jingjing Wang. 2025. Table-critic: A multi-agent framework for collaborative criticism and refinement in table reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17432–17451, Vienna, Austria. Association for Computational Linguistics.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, and 1 others. 2026. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*.

Ruiyi Zhang, Peijia Qin, Qi Cao, Li Zhang, and Pengtao Xie. 2026. Aibuildai: An ai agent that automatically builds ai models.

Jiale Zhao, Guoxin Chen, Fanzhe Meng, Minghao Li, Jie Chen, Hui Xu, Yongshuai Sun, Wayne Xin Zhao, Ruihua Song, Yuan Zhang, and 1 others. 2026. Immersion in the github universe: Scaling coding agents to mastery. *arXiv preprint arXiv:2602.09892*.

Mingyang Zhou, Quanming Yao, Lun Du, Lanning Wei, and Da Zheng. 2025. RePro: Reflective paper-to-code reproduction enabled by fine-grained verification. *arXiv preprint arXiv:2508.16671*.

Xiangqian Zhu, Yilun Cai, Zhihao Liu, Boyu Zheng, Chunyu Wang, Renhong Ye, Jing Chen, Hao Wang, Wei-Chen Wang, Yaowen Zhang, and 1 others. 2026. Toward ultra-long-horizon agentic science: Cognitive accumulation for machine learning engineering. *arXiv preprint arXiv:2601.10402*.

A Additional Related Work

A.1 Automating AI Research

Recent work has rapidly advanced AI research automation. Broadly, this progress spans three complementary directions. First, automated scientific discovery and research-assistant systems study how agents can generate ideas, synthesize literature, run targeted experiments, and draft scientific artifacts (Lu et al., 2024; Yamada et al., 2025; Tang et al., 2025; Schmidgall et al., 2025; Schmidgall and Moor, 2025; Weng et al., 2026; Analemma, 2026; Liu et al., 2026). Second, objective-driven ML engineering agents study iterative propose–implement–evaluate loops under explicit metrics or benchmark objectives (Chan et al., 2025; Jiang et al., 2025; Zhu et al., 2026; Wan et al., 2025; Nam et al., 2025; Chen et al., 2026b; Karpathy, 2026). Third, paper-to-code and reproduction-oriented systems focus on translating research papers into runnable repositories or improving implementation fidelity (Zhou et al., 2025; Jansen et al., 2025; Li et al., 2025b; Seo et al., 2026; Chen et al., 2026a; Fu et al., 2026). Together, these directions establish many ingredients for autonomous AI research. AiScientist builds on this progress by focusing on a more operationally demanding systems question: how to sustain coherent research-engineering progress across paper understanding, implementation, experimentation, and refinement.

A.2 Multi-Agent Coordination and Long-Horizon Continuity

Multi-agent coordination is a common strategy for extending LLM-based problem solving. Frameworks such as CAMEL, MetaGPT, and ChatDev show that role decomposition and structured collaboration can improve complex task solving (Li et al., 2023; Hong et al., 2023; Qian et al., 2024). More recent systems bring related coordination patterns to research-oriented and long-horizon workflows (Schmidgall et al., 2025; Chen et al., 2025; Pu et al., 2025; Yu et al., 2025; Wan et al., 2025; Zhao et al., 2026; Zhu et al., 2026). At the same time, recent analyses suggest that multi-agent systems can fail through brittle handoffs, weak verification, and loss of decision-relevant context (Cemri et al., 2025; Yan et al., 2025; Tang et al., 2026). Our work treats long-horizon performance as a problem of both orchestration and continuity. Rather than relying primarily on conversational handoffs, AiScientist externalizes paper analyses, plans, code-side decisions, and experimental evidence into durable artifacts that downstream agents can repeatedly inspect and build on. This design directly targets the two continuity requirements introduced in the main text: *role continuity*, where later invocations of a specialist resume prior role-level work, and *project continuity*, where different specialists coordinate around shared evidence. In this sense, AiScientist is not simply another hierarchical multi-agent arrangement, but a coordination design for long-horizon ML research engineering centered on artifact-mediated continuity and *thin control over thick state*.

B File-as-Bus Implementation Details

File-as-Bus is implemented as a schema-governed artifact protocol rather than as an unstructured working directory, expanding the method description in Section 3.2 and Figure 1. Each artifact has a declared purpose, update mode, writer set, and reader set. The schema gives each specialist a stable contract: which evidence it should consult before acting, which artifacts it owns, and which downstream roles will consume its outputs. Table 3 summarizes the artifacts used by AiScientist.

The update mode determines how an artifact may evolve. *Versioned artifacts* have a canonical current version while preserving prior revisions conceptually; they are used for analyses and plans whose latest form should be easy to find, but whose history may matter for later debugging. *Append-only logs* preserve chronological evidence and should not be rewritten; they are used for implementation rationales, experiment traces, failures, and diagnoses. *Mutable runnable artifacts* are directly edited project state, such as source code, setup scripts, and execution entry points. Their detailed decision history is preserved through the append-only logs rather than by versioning every file in the runnable repository.

The access policy is role-scoped. Tier-1 specialists write primarily to artifacts corresponding to their responsibility, while all roles can inspect upstream evidence that affects their decisions. Tier-2 subagents are used for bounded subtasks and default to read-only access unless the invoking specialist folds their findings back into the appropriate artifact. After each specialist invocation, the workspace map is refreshed from the current artifact state and schema metadata, giving the Orchestrator a compact index of available evidence

Artifact	Purpose	Update Mode	Writer	Readers
paper_analysis/summary.md	Concise implementation-oriented summary of the paper, including task goal, target result, and key assumptions.	Versioned artifact	Paper Comprehension	Orchestrator, Prioritization, Implementation, Experimentation
paper_analysis/structured.md	Structured extraction of datasets, metrics, methods, baselines, training details, and resource requirements.	Versioned artifact	Paper Comprehension	Orchestrator, Prioritization, Implementation, Experimentation
paper_analysis/algorithm.md	Algorithm-level reconstruction of the proposed method and implementation-relevant details.	Versioned artifact	Paper Comprehension	Implementation, Experimentation, Orchestrator
paper_analysis/baseline.md	Baseline methods, expected comparisons, and result targets that should be reproduced or approximated.	Versioned artifact	Paper Comprehension	Prioritization, Implementation, Experimentation, Orchestrator
agent/prioritized_-task.md	Current ordered task plan, including dependencies, expected impact, feasibility, and next actions.	Versioned artifact	Prioritization	Orchestrator, Implementation, Experimentation
submission/	Runnable repository under construction, including source code, configuration, scripts, and generated project files.	Mutable runnable artifact	Implementation	Implementation, Experimentation, Orchestrator
submission/setup/	Setup and resource-acquisition scripts, including dependency installation, dataset download, and model preparation.	Mutable runnable artifact	Implementation	Experimentation, Orchestrator
submission/reproduce.sh	Canonical entry point used to rerun the current system in evaluation or validation.	Mutable runnable artifact	Implementation	Experimentation, Orchestrator
agent/impl_log.md	Append-only record of implementation decisions, blockers, deviations from paper analysis, and unresolved code-side issues.	Append-only log	Implementation	Orchestrator, Paper Comprehension, Prioritization, Experimentation
agent/exp_log.md	Append-only record of experimental runs, metrics, failures, diagnoses, and evidence used for later refinement.	Append-only log	Experimentation	Orchestrator, Prioritization, Implementation, Paper Comprehension

Table 3: File-as-Bus artifact schema used by AiScientist. Each artifact has an explicit purpose, update mode, writer, and reader set. This schema turns the shared workspace into a collaboration contract rather than an unstructured scratchpad.

without loading the entire workspace into its active context. This is the implementation-level mechanism behind the thin-control interface described in Section 3.3.

C Evaluation and Metric Details

Code and artifacts. We include the anonymized implementation, evaluation scripts, and experiment artifacts in the supplementary material submitted with this paper.

PaperBench. PaperBench (Starace et al., 2025) full evaluation contains 20 paper-replication tasks. For each task, the agent receives the paper specification, a clean execution environment, permitted external-resource access, one H20 GPU, and a 24-hour budget. The agent must build a fresh implementation without using the authors' original code or other blacklisted resources. We follow the official full-evaluation protocol and use GPT-5.4 (OpenAI, 2026) as the grading model. Because a full 20-task evaluation under this grading setup costs approximately $832, repeated full-benchmark evaluations are materially constrained.

MLE-Bench Lite. MLE-Bench Lite (Chan et al., 2025) contains 22 competition-style ML tasks. Each run has a 24-hour budget and one H20 GPU. We report metrics over three runs/seeds following the MLE-Bench convention, with *Any Medal%* as the primary metric. Controlled rows in Table 2 use our evaluation setup, while official leaderboard rows are contextual references reported from the MLE-Bench ecosystem.

Anytime score curves. This definition corresponds to the mean best-so-far score panel in Figure 2. For each wall-clock time point t, let $x_i(t)$ be the validation best-so-far normalized score for valid task–seed trajectory i, and let n_t be the number of valid trajectories at time t. The plotted mean is

$$\bar{x}(t) = \frac{1}{n_t} \sum_{i=1}^{n_t} x_i(t). \tag{5}$$

Let $s(t)$ be the sample standard deviation of $\{x_i(t)\}_{i=1}^{n_t}$. The shaded band is $\bar{x}(t) \pm s(t)/\sqrt{n_t}$, i.e., ± 1 standard error of the mean (SEM) across task–seed trajectories at that time point.

Pairwise lead rate. This definition corresponds to the pairwise lead-rate panel in Figure 2. For each

matched task–seed comparison at time t, we assign

$$w_i(t) = \begin{cases} 1, & \text{if AiScientist leads the reference,} \\ 0.5, & \text{if tied,} \\ 0, & \text{if the reference leads AiScientist.} \end{cases} \tag{6}$$

The lead rate is $\hat{p}(t) = \frac{1}{n}\sum_{i=1}^{n} w_i(t)$. For visualization, shaded bands use the binomial standard-error approximation

$$\sqrt{\frac{\hat{p}(t)(1-\hat{p}(t))}{n}}. \tag{7}$$

This view complements the average score curve: the score curve measures the magnitude of improvement, while the lead-rate curve measures how broadly AiScientist is ahead across matched trajectories.

D Baseline and Reference Details

We distinguish matched baselines from contextual references. Matched baselines are evaluated under the same controlled setup as AiScientist and support direct system-level comparisons. Contextual references provide calibration against strong external systems or official leaderboard results, but are not treated as matched baselines because they may differ in model, harness, or reporting protocol.

On PaperBench, BasicAgent and IterativeAgent are the matched baselines from the official benchmark protocol (Starace et al., 2025). They are evaluated under the same task suite and grading setup as AiScientist. Codex with GPT-5.5 xhigh is reported as a frontier harness reference (OpenAI, 2025): it is useful for calibrating against a strong contemporary coding setup, but it is not a matched baseline because both the harness and backbone model differ from our controlled comparisons.

On MLE-Bench Lite, AIDE (Jiang et al., 2025), LoongFlow (Wan et al., 2025), and ML-Master 2.0 (Zhu et al., 2026) are controlled comparison systems in our setup. Official MLE-Bench Lite leaderboard rows are included as contextual references to place the controlled results in the broader ecosystem (Yang et al., 2025; Li et al., 2025a; Liu et al., 2025; Toledo et al., 2025; Nadafian et al., 2026; Chen et al., 2026b; Zhang et al., 2026). The Codex/GPT-5.5 xhigh row is again a frontier harness reference evaluated under our setup, intended to answer how AiScientist compares with a strong black-box coding harness rather than to isolate the effect of a single architectural choice.

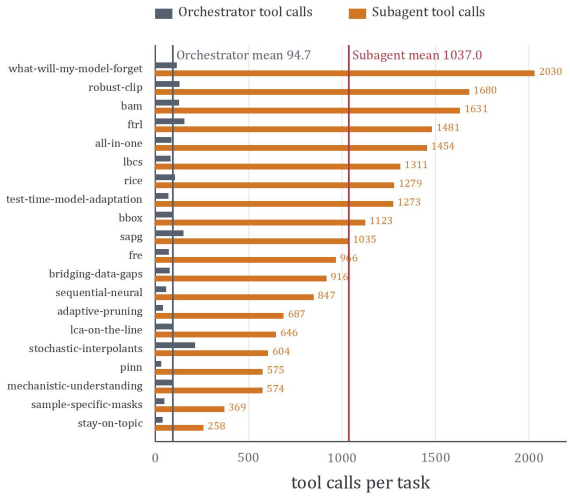


Figure 6: Delegation ratio across PaperBench tasks under GLM-5. Subagent delegation is not concentrated in a small number of tasks; it appears across most tasks, indicating that the hierarchical team is used as a systematic control mechanism rather than as an occasional fallback.

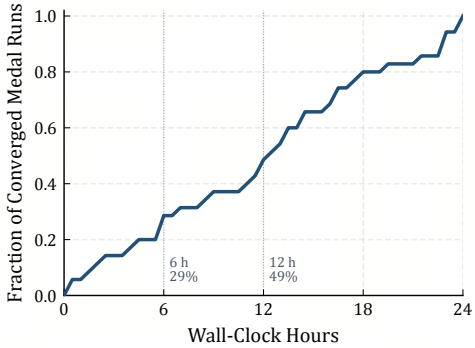


Figure 7: Budget-dependent medal outcomes on MLE-Bench Lite under GLM-5. The figure provides a supplementary view of how competitive outcomes emerge over the 24-hour budget.

E Supplementary Analyses

E.1 Delegation Patterns on PaperBench

Figure 6 complements the workflow-step analysis in Section 4.6, especially Figure 4. The result shows that AiScientist regularly invokes specialist and subagent work across PaperBench tasks. This is consistent with the intended thin-control design: the Orchestrator maintains stage-level control, while substantial local work is delegated to focused agents whose outputs are folded back into the shared workspace.

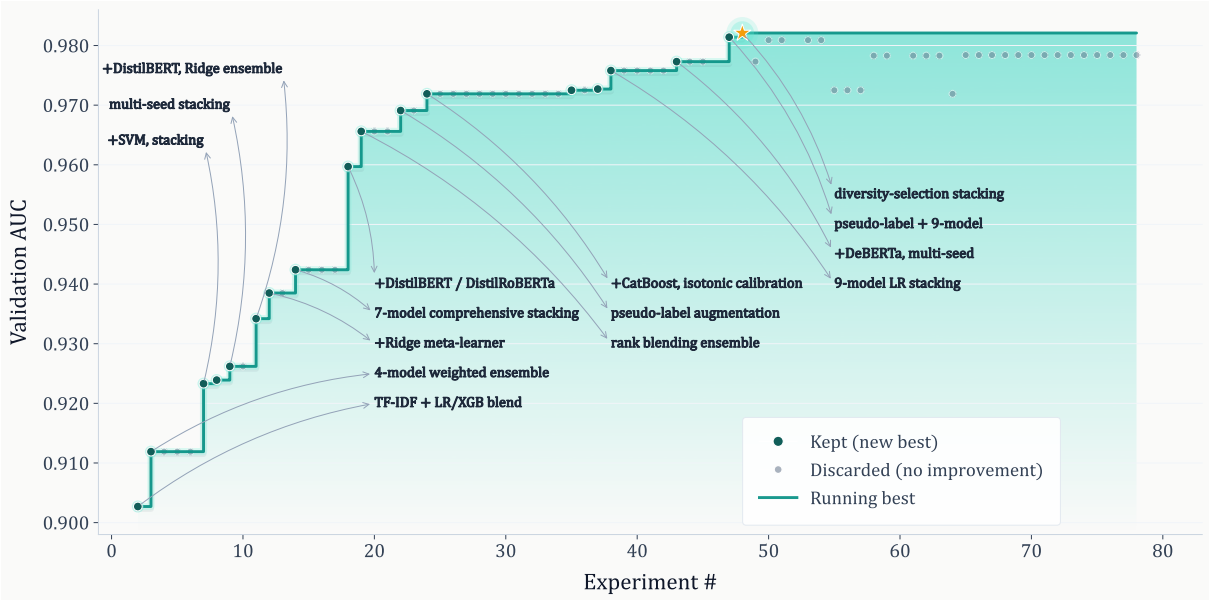


Figure 8: AiScientist autonomously improves performance on a competition-style ML task over 23 hours. On MLE-Bench Lite's *Detecting Insults* task, it conducts 74 experiment cycles without human intervention, raising validation AUC from 0.903 to 0.982 through 18 best-so-far updates. This case illustrates how repeated experimentation and durable experiment records support continued refinement within a single task.

E.2 Budget and Medal Outcomes on MLE-Bench Lite

Figure 7 provides a complementary view of long-horizon behavior on MLE-Bench Lite. Where Figure 2 focuses on score trajectories, pairwise lead rates, and time to final best score, this analysis summarizes how medal-level outcomes depend on the available budget. It supports the same interpretation as the main dynamics figure: later-budget refinement is important for converting runnable submissions into stronger competitive outcomes.

E.3 Case Study: Detecting Insults

Figure 8 illustrates the behavior behind the aggregate long-horizon trends in Figure 2. On a single competition-style task, AiScientist continues to improve after the initial runnable solution, using repeated experiment cycles to refine the model and validation performance. The case is not intended as an additional benchmark result; rather, it shows qualitatively how the evidence-driven loop operates within one task trajectory.