AIS-BENCH: AN AI SCIENTIST BENCHMARK FOR AUTONOMOUS SCIENTIFIC DISCOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

The emergence of AI Scientists—agents that generate hypotheses, implement methods, and report findings—calls for evaluation beyond isolated tasks or final manuscripts. Existing benchmarks emphasize individual capabilities or end-to-end outcomes, leaving the fidelity of intermediate scientific artifacts underexamined. We introduce **AIS-Bench**, a diagnostic framework that evaluates automated research through three independently scored phases: **Reference-Conditioned Idea Reconstruction**, **Code Implementation**, and **Controlled Reporting**. From 3,174 machine-learning papers, our construction protocol yields 753 aligned $\langle \text{Context}, \text{Idea}, \text{Code}, \text{Paper} \rangle$ quadruplets with executable reference implementations. We evaluate 13 prompting strategies, specialized systems, and end-to-end agents using phase-specific metrics, with judge-based components validated against held-out human judgments. On matched quadruplets under a common human-calibrated artifact-validity criterion, implementation receives the lowest phase score; paired bootstrap intervals and alternative aggregations preserve this ordering. Audits further show that functional-logic deviations dominate classified implementation failures, while metric exaggeration and unresolved citations persist in reports despite specified evidence. AIS-Bench enables systematic, traceable diagnosis of scientific-fidelity degradation across the research lifecycle and supports the development of verification-centered AI Scientists.

1 INTRODUCTION

AI Scientists are becoming increasingly important for scaling scientific discovery because they integrate research activities otherwise automated in isolation. Recent frameworks link hypothesis formation, experimental implementation, and reporting into sustained workflows (Lu et al., 2024; Yamada et al., 2025; Gottweis et al., 2025; Tang et al., 2025). *DeepScientist* (Weng et al., 2026) investigates iterative research, while *DeepResearcher* (Zheng et al., 2025) supports web-based information synthesis. Such integration is particularly valuable when progress depends on repeated transitions among conceptual reasoning, software execution, evidence interpretation, and experimental revision. The resulting capacity for long-horizon reasoning and tool use could accelerate the conversion of scientific questions into testable hypotheses and verifiable evidence. Rigorous evaluation is essential because errors in intermediate artifacts can propagate into executable results and apparently coherent conclusions.

Current AI Scientists can produce plausible artifacts despite invalid ideas, code that deviates from intended mechanisms, or reports with overstated results and unresolved references. End-to-end scores obscure the source of these failures. Existing benchmarks provide partial views: MLAgentBench and MLE-Bench assess executable ML engineering (Huang et al., 2023; Chan et al., 2025); PaperBench evaluates research replication (Starace et al., 2025); and MLR-Bench covers stepwise and end-to-end outputs (Chen et al., 2025a). ResearchClawBench and AARRI-Bench cover longer-horizon research (Xu et al., 2026; Wang et al., 2026), while AiScientist studies matched-backbone agents (Chen et al., 2026). However, corpus-scale audits of aligned idea, code, and report artifacts from the same trajectory remain absent. This leaves a central question: *Can AI Scientists preserve scientific fidelity from idea to executable code and evidence-grounded reporting?*

To address these limitations, we introduce **AIS-Bench**, spanning *Reference-Conditioned Idea Reconstruction*, *Code Implementation*, and *Controlled Reporting*. From 3,174 peer-reviewed

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

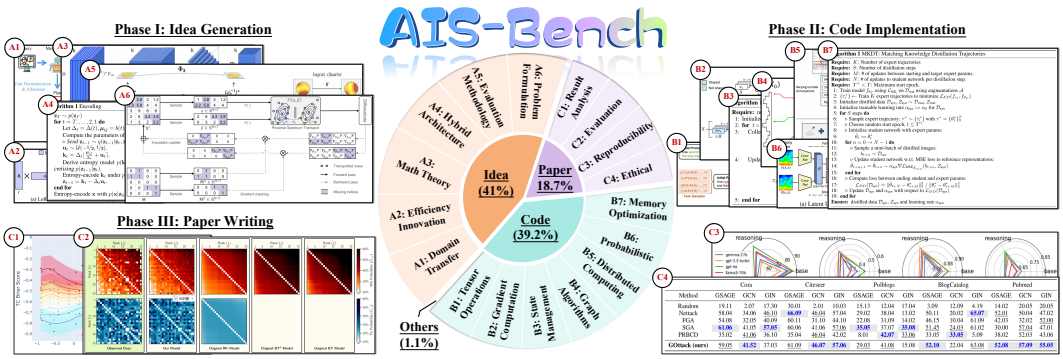

Figure 1: **Distribution of the AIS-Bench Dataset.** The central chart summarizes 17 diagnostic subsets across the Idea, Code, and Paper phases, plus an *Other* category. Surrounding panels are schematic views of the three artifact types; callouts index taxonomy labels rather than paper-level annotations. Appendix Table 6 provides the operational definitions.

machine-learning papers, we derive 753 aligned $\langle \text{Context}, \text{Idea}, \text{Code}, \text{Paper} \rangle$ quadruplets with executable code, standardized inputs, and auditable references. Across 13 configurations, matched-quadruplet analyses identify implementation fidelity as the central bottleneck. Independently assigned taxonomy labels, held-out judge validation, and sample-level records support stage-specific diagnosis (Figure 1). Our main contributions are summarized as follows:

- We introduce **AIS-Bench**, an AI Scientist benchmark for autonomous scientific discovery that decouples reference-conditioned idea reconstruction, code implementation, and controlled reporting using standardized interfaces and phase-specific metrics.
- We curate a corpus of **3,174** papers and an executable core of **753** aligned quadruplets, organized using an operational taxonomy with independently assigned labels and expert adjudication.
- Across 13 configurations, we identify a persistent code-score deficit and characterize dominant engineering and reporting failures using matched comparisons, uncertainty estimates, and sample-level audits. These findings motivate verification methods linking intended mechanisms, executable behavior, and reported evidence.

2 Related Work

AI Scientists and Research Agents. Automated scientific discovery spans domain-specific tools (Shen et al., 2021) and scientific-reasoning benchmarks (Wang et al., 2023), and now includes LLM agents that coordinate research activities. *Co-Scientist* (Gottweis et al., 2025) generates and ranks biomedical hypotheses, whereas *The AI Scientist* (Lu et al., 2024) orchestrates an ML research workflow. *AI-Researcher* (Tang et al., 2025) targets integrated scientific innovation, while *DeepResearcher* (Zheng et al., 2025) develops web-research agents through reinforcement learning for evidence gathering and synthesis. These systems differ in lifecycle coverage and artifacts, motivating evaluations that separate generation from verification and test cross-stage consistency.

Benchmarks for Automated Science. As summarized in Table 1, representative benchmarks cover idea generation and data-driven discovery (Majumder et al., 2025; Guo et al., 2025; Qiu et al., 2025; Chen et al., 2025b), literature retrieval and synthesis (Du et al., 2026; Shao et al., 2026), executable experimentation (Huang et al., 2023; Chan et al., 2025; Starace et al., 2025; Kon et al., 2026), or multiple lifecycle stages (Zhang et al., 2025; Chen et al., 2025a). Recent ResearchClawBench and AARRI-Bench assess long-horizon research (Xu et al., 2026; Wang et al., 2026), while AiScientist controls model backbones in engineering-agent comparisons (Chen et al., 2026). Yet corpus-scale auditing of aligned idea, code, and report artifacts remains limited. AIS-Bench addresses this gap by aligning stage inputs, outputs, and references to a common paper-derived trajectory with executable code and execution-grounded evidence.

Table 1: **Comparison with representative benchmarks.** AIS-Bench combines dynamic execution, separately scored stages, and aligned *Idea–Code–Paper* artifacts. **Dyn.**: execution; **Dec.**: separate stage scores; **Align.**: same-trajectory artifacts. See Appendix A for definitions and sources.

Benchmark	Lifecycle Scope			Methodology				Dataset Stats	
	Idea	Code	Paper	Dyn.	Dec.	Align.	Source Type	Size	Domain
ScienceAgentBench	✗	✓	✗	✓	✗	✗	Research Papers	102 Tasks	General
PaperBench	✗	✓	✗	✓	✗	✗	Research Papers (ICML 2024)	20 Papers (8,316 Rubric Items)	ML
IdeaBench	✓	✗	✗	✗	✗	✗	Research Papers	2,374 (Target Papers)	General
AI Idea Bench 2025	✓	✗	✗	✗	✗	✗	Research Papers	3,495	ML
SciNetBench	✗	✗	✓	✗	✗	✗	Research Papers	1,087 Queries (18.6M Corpus)	AI
DeepResearch Bench	✗	✗	✓	✗	✗	✗	Expert Crafted	100 Tasks	22 Fields
Auto-Bench	✓	✗	✗	✓	✗	✗	Synthetic (Causal Graphs)	Procedural	General
MLAgentBench	✓	✓	✗	✓	✗	✗	Kaggle, Research Datasets	13 Tasks	ML
DiscoveryBench	✓	✓	✗	✓	✓	✗	Research Papers, Synthetic	1,167 (264 Real + 903 Synth)	General
MLE-Bench	✗	✓	✗	✓	✗	✗	Kaggle Competitions	75 Competitions	ML
HiSciBench	✓	✓	✓	✓	✓	✗	Research Papers, Public Datasets	8,735	General
EXP-Bench	✓	✓	✗	✓	✗	✗	Research Papers (Top-tier AI)	461 Tasks	ML
MLR-Bench	✓	✓	✓	✓	✓	✗	Workshop Topics	201 Tasks	ML
AIS-Bench	✓	✓	✓	✓	✓	✓	**Research Papers**	**3,174 (753 Core)**	**ML**

3 AIS-BENCH: BENCHMARK FOR AI SCIENTISTS

AIS-Bench first constructs aligned scientific quadruplets and then evaluates agent outputs through three independently scored phases (Figure 2).

3.1 THE SCI-QUADRUPLET CONSTRUCTION PROTOCOL

We define a benchmark instance as a quadruplet $\mathcal{Q} = \langle \mathcal{X}, \mathcal{I}, \mathcal{C}, \mathcal{P} \rangle$, where $\mathcal{X}$ denotes the sanitized research background and problem formulation, $\mathcal{I}$ the structured scientific hypothesis, $\mathcal{C}$ the executable reference implementation, and $\mathcal{P}$ the source manuscript. We write $\mathcal{I}_{GT} \equiv \mathcal{I}$ and $\mathcal{C}^{GT} \equiv \mathcal{C}$ when emphasizing their roles as references. The manuscript is retained for offline provenance and auditing, whereas each evaluation phase receives controlled agent-facing inputs; Phase III is scored against execution evidence. Construction stages create aligned artifacts from paper–repository pairs, while evaluation phases score agent outputs. We construct the quadruplets through three stages:

Stage 1: Sourcing & Executability Filtering. We collect 3,174 papers from ICLR, ICML, and NeurIPS (2023–2025) through the OpenReview API[1] and cross-reference them with Papers with Code[2]. To establish standardized executability, we apply two filters: *1) Dependency Resolution:* We combine repository manifests with pipreqs-style[3] import analysis to resolve dependencies in isolated Docker environments. *2) Reference Executability Check:* We retain repositories whose documented entry point completes a one-epoch sandbox run without runtime errors. This yields an executable core of 753 aligned quadruplets.

Stage 2: Idea Distillation. We represent each scientific idea as a structured tuple $\mathcal{I}$ containing its motivation, mechanism, and intended outcome. GPT-4o extracts these fields from the methodology section under the constraints in Protocol A of Appendix H.1, preserving algorithmic logic while excluding implementation details. Human annotators audit a stratified set of 75 samples (approximately 10% of the executable core) to assess whether each tuple preserves the source method's core technical contribution.

Stage 3: Context Generation. The context ($\mathcal{X}$) defines a reference-conditioned reconstruction setting. We combine the introduction with cited prior-work context, apply Named Entity Masking (NEM), and replace specific method names with `[OUR_METHOD]` (Appendix H.2). We remove residual solution-revealing cues identified through exact- and near-duplicate matching. Temporal-holdout and similarity-stratified evaluations assess robustness across publication periods and estimated exposure strata.

[1] https://openreview.net/about/api
[2] https://paperswithcode.com/
[3] https://github.com/bndr/pipreqs

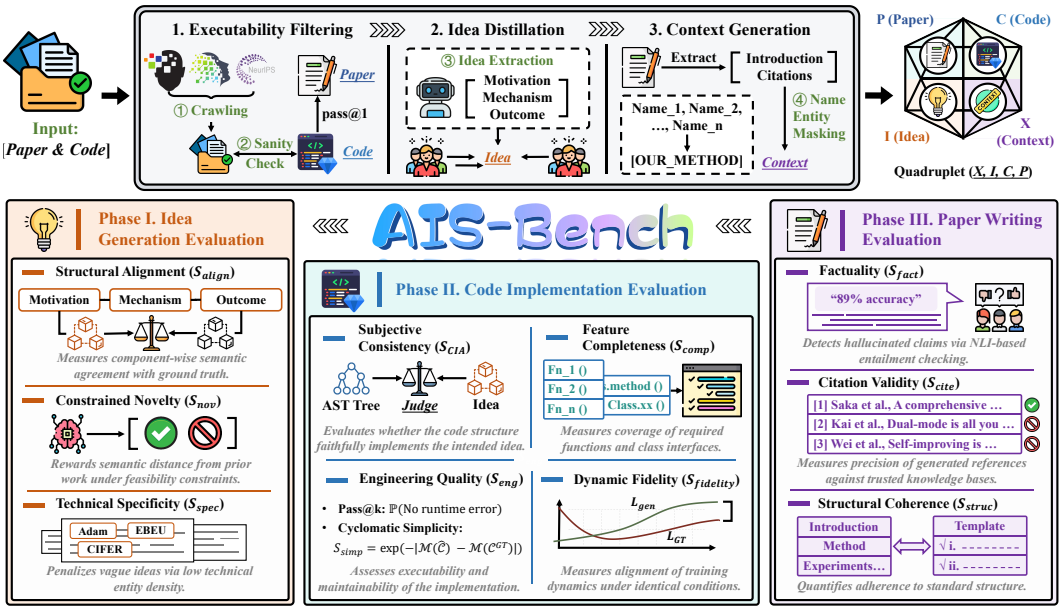

Figure 2: **Overview of the AIS-Bench Framework.** Figure labels denote Reference-Conditioned Idea Reconstruction (Idea Generation), reference-entity overlap (Technical Specificity), Controlled Reporting (Paper Writing), IMRaD compliance (Structural Coherence), and cyclomatic similarity (maintainability); Pass@k uses $k = 1$.

3.2 Phase I: Reference-Conditioned Idea Reconstruction

Given $f_\theta : \mathcal{X} \to \hat{\mathcal{I}}$, the agent reconstructs a structured hypothesis $\hat{\mathcal{I}}$ from the sanitized target-paper context. The task evaluates faithful recovery of the reference idea while requiring a technically sound formulation. Our protocol combines a calibrated LLM judge, embedding-based distance, and reference-entity overlap along three dimensions (Appendix I.4):

Idea-1: Structural Alignment (S_{align}). To measure semantic agreement with the reference trajectory, the judge assigns each component a 1–5 rating, linearly normalized to $[0, 1]$. Alignment is the mean of these normalized component-wise similarities:

$$S_{align} = \sum_{k \in \mathcal{K}} w_k \cdot \text{Sim}_{judge}(\hat{\mathcal{I}}_k, \mathcal{I}_{GT,k}) \tag{1}$$

where $\mathcal{K} = \{\text{Motivation}, \text{Mechanism}, \text{Outcome}\}$ and $w_k = 1/|\mathcal{K}|$. The Sim_{judge} function returns a score in $[0, 1]$ quantifying semantic and technical agreement for each component.

Idea-2: Constrained Novelty (S_{nov}). We quantify novelty as the semantic distance to a leave-one-out knowledge base $\mathcal{D}_{retrieval}$ that excludes the target paper and duplicate versions. A binary feasibility gate ($\mathbb{I}_{feas}$) restricts credit to proposals judged technically sound:

$$S_{nov} = \mathbb{I}_{feas}(\hat{\mathcal{I}}) \cdot \min_{d \in \mathcal{D}_{retrieval}} \text{clip}_{[0,1]}\big(1 - \cos(\mathbf{e}_{\hat{\mathcal{I}}}, \mathbf{e}_d)\big) \tag{2}$$

where the judge sets $\mathbb{I}_{feas} = 1$ for proposals assessed as consistent with fundamental ML principles and 0 otherwise.

Idea-3: Reference Specificity (S_{spec}). Using named entity recognition (NER), we measure whether the reconstruction preserves reference-specific technical entities. Let $\mathcal{E}_{gen}$ and $\mathcal{E}_{GT}$ be entities from $\hat{\mathcal{I}}$ and $\mathcal{I}_{GT}$. We compute $S_{spec} = |\mathcal{E}_{gen} \cap \mathcal{E}_{GT}|/|\mathcal{E}_{gen} \cup \mathcal{E}_{GT}|$, assigning zero when the union is empty. The resulting overlap operationalizes technical specificity relative to the reference idea.

3.3 Phase II: Code Implementation Evaluation

The agent translates the structured reference hypothesis into an implementation $\hat{\mathcal{C}} = g_\theta(\mathcal{I}_{GT})$. Supplying the reference idea isolates implementation behavior from upstream reconstruction errors. We evaluate fidelity using code-structure analysis and dynamic execution (Appendix J.1):

Code-1: Code–Idea Alignment (S_{CIA}). To detect architectural deviations that syntax checks miss, we compute the Code–Idea Alignment (CIA) score. We parse $\hat{\mathcal{C}}$ into an Abstract Syntax Tree (AST) summary containing key definitions and algorithmic loops. A judge then reviews this summary against $\mathcal{I}_{GT}$:

$$r_{CIA} = \text{Judge}(\text{AST}(\hat{\mathcal{C}}), \mathcal{I}_{GT}), \qquad S_{CIA} = \frac{r_{CIA} - 1}{4}, \tag{3}$$

where $r_{CIA} \in \{1, \ldots, 5\}$, so $S_{CIA} \in [0, 1]$.

Code-2: Feature Completeness (S_{comp}). We estimate coverage of reference-defined functional blocks by comparing code symbol tables. Let $\mathcal{F}_{GT}$ be the critical function signatures and class methods in $\mathcal{C}^{GT}$. Their recall is

$$S_{comp} = \frac{|\text{Matched}(\text{Symbols}(\hat{\mathcal{C}}), \mathcal{F}_{GT})|}{|\mathcal{F}_{GT}|} \tag{4}$$

Code-3: Engineering Quality (S_{eng}). We combine strict executability with a lightweight structural-similarity proxy:

- **Strict Pass@1:** $\text{Pass@1} = 1$ when one execution completes within the sandbox budget with exit code 0 and no import, syntax, runtime, or resource failure; otherwise it is 0.
- **Cyclomatic Similarity:** We compare McCabe's cyclomatic complexity $\mathcal{M}$ with the reference: $S_{simp} = \exp(-|\mathcal{M}(\hat{\mathcal{C}}) - \mathcal{M}(\mathcal{C}^{GT})|)$. This is a structural proxy, not an independent measure of maintainability; $S_{eng} = (\text{Pass@1} + S_{simp})/2$.

Code-4: Dynamic Fidelity ($S_{fidelity}$). Execution success is necessary but not sufficient. We compare early optimization behavior by running $\hat{\mathcal{C}}$ and $\mathcal{C}^{GT}$ with fixed random seeds on identical synthetic data for $T = 50$ steps:

$$S_{fidelity} = \exp\Big(-\frac{1}{T} \sum_{t=1}^{T} \frac{|\mathcal{L}_t^{gen} - \mathcal{L}_t^{GT}|}{|\mathcal{L}_t^{GT}| + \epsilon} \Big) \tag{5}$$

Here $\epsilon > 0$ is fixed across systems and stabilizes the denominator. Higher $S_{\text{fidelity}} \in [0, 1]$ indicates closer pointwise agreement between the first T loss values; failed executions receive $S_{\text{fidelity}} = 0$.

3.4 Phase III: Controlled Reporting

Given $h_\theta : \mathcal{E}_{report} \to \hat{\mathcal{P}}$, the agent receives a controlled evidence bundle containing the sanitized problem context, distilled method card, reference implementation, execution logs, and a retrievable literature index. The source manuscript $\mathcal{P}$ remains withheld. The agent produces an IMRaD-aligned report with a bibliography, enabling direct audits of result-claim support, bibliographic validity, and structural compliance (Appendix K.1):

Paper-1: Result-Claim Factuality (S_{fact}). We cast verification of numerical and comparative claims as a Natural Language Inference (NLI) task. An information extraction pipeline isolates atomic claims $\mathcal{S}$ containing metrics or trends and tests each claim against $\mathcal{L}_{exec}$:

$$S_{fact} = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \mathbb{I}[\text{NLI}(\mathcal{L}_{exec}, s) = \text{entailed}] \tag{6}$$

We assign $S_{fact} = 0$ when no auditable claim is extracted, preventing omission from receiving a perfect score. Performance-overstating contradictions are labeled metric exaggerations; claims about absent experiments are treated as unsupported.

Paper-2: Citation Validity (S_{cite}). We verify generated references $\mathcal{R}_{gen}$ against S2ORC, arXiv, and DBLP. The automatic score measures bibliographic existence:

$$S_{cite} = \frac{|\{r \in \mathcal{R}_{gen} \mid \text{Exists}(r, \mathcal{D}_{KB})\}|}{|\mathcal{R}_{gen}|} \tag{7}$$

We assign $S_{cite} = 0$ when $\mathcal{R}_{gen}$ is empty. This metric measures bibliographic resolvability, not topical or evidential relevance.

Paper-3: IMRaD Structural Compliance (S_{struc}). A fixed discourse parser maps the controlled report to a sequence of rhetorical moves $\mathrm{Seq}(\hat{\mathcal{P}})$. We compare this sequence with the requested IMRaD template T_{IMRaD} using normalized Levenshtein distance:

$$S_{struc} = 1 - \frac{\mathrm{Levenshtein}(\mathrm{Seq}(\hat{\mathcal{P}}), \mathrm{T}_{IMRaD})}{\max(|\,\mathrm{Seq}(\hat{\mathcal{P}})|, |\mathrm{T}_{IMRaD}|)}. \tag{8}$$

This score measures requested-template compliance rather than general writing quality or scientific validity.

Reported phase scores. All component metrics are normalized to $[0, 1]$ and combined using equal-weight phase composites. Table 3 reports mean subset scores and four-subset macro-averages on a 0–100 scale. Appendix K.3 defines aggregation; the anonymous artifact provides component results, alternative-weight analyses, and a Code composite gated by Strict Pass@1.

4 Experiments

We empirically evaluate LLMs and agent frameworks through three research questions:

- **RQ1 (Capability Profile):** How do systems perform across reference-conditioned idea reconstruction, code implementation, and controlled reporting?
- **RQ2 (Engineering Failure Composition):** How are failed code-generation runs distributed across engineering failure categories?
- **RQ3 (Epistemic Integrity):** To what extent do agents produce unsupported result claims or unresolved citations during controlled reporting?

4.1 Experimental Setup

Datasets. We construct **AIS-Bench** from **3,174** peer-reviewed papers collected from ICLR[4], NeurIPS[5], and ICML[6] (2023–2025), retaining **753** samples with executable reference code for phase-wise evaluation. Table 2 distinguishes these populations: the full corpus supports taxonomy construction and distributional analysis, whereas the executable core provides the evaluation pool. **Level 1** classifies each paper's primary contribution and is dominated by *Methodological Proposals* (87.9%). **Level 2** assigns each paper to one of 17 diagnostic categories spanning Idea Reconstruction, Code Implementation, and Controlled Reporting; 36 papers outside these categories are grouped as *Other*. Table 3 reports the four most prevalent full-corpus categories in each phase and evaluates their corresponding executable-core subsets. Figure 1 visualizes the taxonomy, with definitions in Appendix B.

Table 2: **Statistics of the AIS-Bench Dataset.** Level 1 summarizes article types, while Level 2 reports content-based diagnostic categories.

Category	Count (Ratio)
Global Scale	
Full Corpus (Taxonomy)	**3,174**
Executable Core (Evaluation)	**753** (23.7%)
Level 1: Article Type Distribution	
Methodological Proposal	2,791 (87.9%)
Theoretical Analysis	221 (7.0%)
Benchmark / Dataset	134 (4.2%)
Empirical Discovery	28 (0.9%)
Level 2: Diagnostic Category Distribution	
Phase I: Idea Reconstruction (A1–A6)	1,300 (41.0%)
Phase II: Code Implementation (B1–B7)	1,245 (39.2%)
Phase III: Controlled Reporting (C1–C4)	593 (18.7%)
Other	36 (1.1%)

Baselines. We evaluate 13 configurations in three groups. **1) Prompting Strategies:** Zero-shot, few-shot, and chain-of-thought prompting use Claude 3.5 Sonnet without an agent scaffold. **2) Specialized Systems:** Phase-specific baselines comprise *SciMON* (Wang et al., 2024) and *ResearchAgent* (Baek et al., 2025) for ideation, *DeepCode* (Li et al., 2025) and *CodeScientist* (Jansen et al., 2025) for coding, and *CycleResearcher* (Weng et al., 2025) and *Agent Laboratory* (Schmidgall et al., 2025) for writing. **3) End-to-End Autonomous Scientists:** We adapt *AI Scientist (v1)* (Lu et al., 2024), *AI Scientist (v2)* (Yamada et al., 2025), *DeepResearcher* (Zheng et al., 2025), and *AI-Researcher* (Tang et al., 2025) to the benchmark's phase-specific interfaces; the artifact documents each adaptation and its native configuration.

[4] https://iclr.cc
[5] https://neurips.cc
[6] https://icml.cc

Table 3: **Main Results on Selected Phase Subsets (Core $N = 753$).** Mean composite scores (0–100) use the four most prevalent categories per phase (A4/A5/A3/A6, B7/B3/B4/B1, C4/C3/C1/C2). The anonymous artifact provides per-column n, sample IDs, bootstrap 95% CIs, and all-category results. Avg. is the four-subset macro-average; **bold** marks group best, underline column best, and "–" unevaluated phases.

System	Phase I: Idea Reconstruction (0–100)					Phase II: Code Implementation (0–100)					Phase III: Controlled Reporting (0–100)				
	Hybr.	Eval.	Theo.	Form.	Avg	Mem.	State	Graph	Tens.	Avg	Ethic.	Repro.	Res.	Comp.	Avg
1. Prompting Strategies (Base: Claude 3.5 Sonnet)															
Zero-Shot	12.5	10.0	6.5	11.5	10.1	4.2	1.5	3.0	7.5	4.1	18.5	8.0	20.5	15.0	15.5
Few-Shot	18.0	15.5	9.0	16.5	14.8	7.5	3.0	5.5	10.5	6.6	28.0	14.5	32.0	24.5	24.8
Chain-of-Thought	**24.5**	**20.0**	**13.5**	**22.0**	**20.0**	**10.5**	**5.5**	**8.0**	**14.0**	**9.5**	**36.5**	**22.0**	**39.5**	**30.0**	**32.0**
2. Specialized Systems (Phase-Specific)															
Idea: SciMON	68.0	**72.5**	60.5	70.0	67.8	–	–	–	–	–	–	–	–	–	–
Idea: ResearchAgent	**71.5**	69.0	**64.0**	**72.5**	**69.3**	–	–	–	–	–	–	–	–	–	–
Code: DeepCode	–	–	–	–	–	38.5	35.0	**42.5**	41.0	39.3	–	–	–	–	–
Code: CodeScientist	–	–	–	–	–	**43.0**	**38.5**	39.0	**44.5**	**41.3**	–	–	–	–	–
Paper: CycleResearcher	–	–	–	–	–	–	–	–	–	–	76.5	72.0	**79.5**	75.0	75.8
Paper: Agent Laboratory	–	–	–	–	–	–	–	–	–	–	**78.0**	**75.5**	76.0	**78.5**	**77.0**
3. End-to-End Autonomous Scientists															
AI Scientist (v1)	52.5	48.0	38.5	45.0	46.0	15.0	12.5	14.0	18.5	15.0	58.0	45.0	60.5	55.0	54.6
AI Scientist (v2)	60.0	58.5	45.0	56.5	55.0	24.5	20.0	22.5	28.0	23.8	68.5	58.0	70.0	65.5	65.5
DeepResearcher	68.5	65.0	58.0	66.0	64.4	**34.0**	30.5	**33.0**	**37.5**	**33.8**	75.0	68.5	76.0	72.0	72.9
AI-Researcher	**70.0**	**68.5**	**62.5**	**71.0**	**68.0**	31.6	**32.5**	30.8	36.7	32.9	**81.5**	**74.0**	**79.0**	**77.5**	**78.0**

Implementation Details. We use **Claude 3.5 Sonnet** (Anthropic, 2024) as a temperature-zero evaluator for idea–reference alignment, code–idea consistency, and report-claim entailment. To limit training-data contamination by benchmark papers, we prioritize model snapshots with documented knowledge cutoffs and verify results through temporal holdouts restricted to publications appearing after those cutoffs; undisclosed proprietary corpora preclude guarantees. Instances come from the executable core ($N = 753$), with each column using its diagnostic subset. Metrics are normalized to $[0, 1]$, aggregated as defined in Appendix K.3, and reported on a 0–100 scale. On a disjoint 75-sample test set, judge–human Pearson correlations exceed 0.88; the artifact provides per-metric estimates and bootstrap 95% confidence intervals. GPT-4o cross-evaluation measures sensitivity to the evaluator and preserves principal within-phase comparisons. The artifact records evaluator, budget, execution, sample-assignment, and judge-output details.

4.2 Main Results

Table 3 reports native phase-composite profiles. Because the composites operationalize different constructs, their absolute scales are descriptive rather than directly commensurate. Across configurations evaluated in all phases, Code is numerically lowest. Among phase-specific systems, *CodeScientist*, the strongest code specialist, reaches 41.3, while the best E2E configuration reaches 33.8. Cross-phase inference instead comes from matched quadruplets: under a common human-calibrated artifact-validity criterion, implementation scores below reconstruction and reporting, with paired bootstrap intervals for the phase differences supporting this ordering. Within phases, the specialist–E2E margins are descriptive. *AI-Researcher* trails *ResearchAgent* by 1.3 points in Idea (68.0 versus 69.3), the adapted *DeepResearcher* trails *CodeScientist* by 7.5 points in Code (33.8 versus 41.3), and *AI-Researcher* exceeds *Agent Laboratory* by 1.0 point in Reporting (78.0 versus 77.0). The largest specialist margin therefore occurs in implementation. Because native configurations differ in models, tools, and scaffolds, these margins do not isolate a scaffold effect; matched-backbone and budget-controlled analyses nevertheless preserve the principal within-phase comparisons. Selected subsets localize the Code profile further. State Management and Graph Algorithms yield comparatively low E2E scores, consistent with their dependence on preserving operational state and structural dependencies. Prompting alone offers limited gains: Chain-of-Thought improves every phase average over Zero-Shot, yet reaches only 9.5 in Code. Code specialists perform better under their evaluated configurations, and component-level, alternative-weight, and Strict-Pass@1-gated analyses preserve the implementation finding. Collectively, the results motivate verification of whether generated programs realize the intended scientific operations throughout execution.

Answer to RQ1: Phase-Specific Score Profile

On matched samples, Code is the lowest-scoring phase under the common human criterion; paired bootstrap intervals for phase differences and alternative aggregations support the same profile, indicating that the implementation deficit is not aggregation-specific.

4.3 Phase I (Idea): The Novelty–Alignment Association

Figure 3 summarizes the novelty–alignment profiles of *ResearchAgent* and *AI-Researcher* across the 300 reported trials. Both fitted curves show an overall negative association, with the specialist retaining higher structural alignment throughout the plotted range. The gap is already visible below 0.7 and becomes more pronounced across the intermediate (0.7–0.85) and high-novelty (> 0.85) intervals. Operationally, the region labeled "Hallucination Tax" measures the reference-alignment cost associated with increasingly divergent proposals; the novelty gate assesses scientific feasibility independently. In this reference-conditioned task, a proposal may depart from the published mechanism while remaining technically plausible. The output-level analysis therefore characterizes an association between exploratory variation and faithful reconstruction rather than a causal trade-off. For both systems, higher-novelty outputs warrant component-level inspection to determine whether the reference motivation, mechanism, and outcome are preserved. This motivates refinement objectives that track structural alignment jointly with novelty and feasibility, making the source of each deviation explicit.

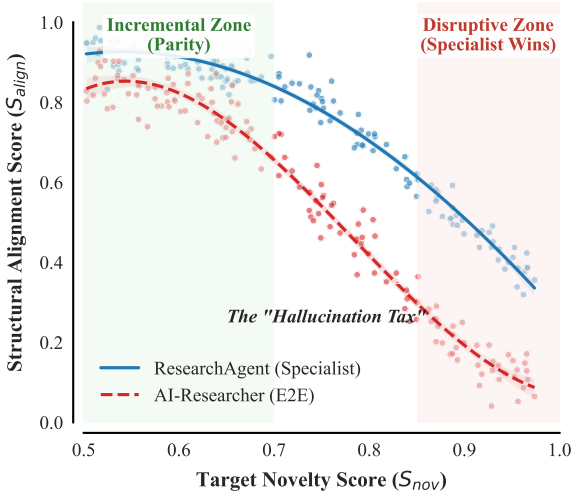

Figure 3: **Novelty–Alignment Association.** Fitted trends across systems; "Target" denotes generated-output novelty, and shading delimits intervals.

4.4 Phase II (Code): Failure Composition

We condition this analysis on Phase-II runs that fail either execution or functional verification and assign each run one mutually exclusive engineering label. Table 4 reports the resulting distribution: functional logic is the largest category for every system, accounting for 43.5–58.8% of classified failures. For CodeScientist, syntax and API issues represent 0.6% and 1.2%, whereas dimension mismatches and functional logic account for 42.5% and 50.2%. Among E2E configurations, AI-Researcher has the smallest dimension-error share (28.5%) and the largest logic-error share (58.8%). Because these are conditional shares, they neither estimate absolute failure rates nor imply that a system with a larger logic share fails more often; the artifact provides absolute counts and run-level outcomes. The functional-logic category includes mechanism deviations that may survive execution, such as substituting standard attention for the requested sparse variant. Across systems, dimension and logic errors together dominate classified failures. This pattern suggests that execution feedback should inspect intermediate tensor states and scientific invariants in addition to exit status.

Answer to RQ2: Failure Beyond Execution

Among runs that fail execution or functional verification, functional-logic deviations dominate for every system. Because some survive execution, evaluating implementation fidelity requires reference-grounded verification beyond conventional debugging.

Table 4: **Engineering failure composition (%).** Conditional shares among failed runs; shading marks each system's dominant category.

System	Execution Failures (%)				Func. (%)
	Syn.	API	Dim.	OOM	Logic
Code Specialists					
DeepCode	1.5	2.2	41.0	8.5	46.8
CodeScientist	0.6	1.2	42.5	5.5	50.2
End-to-End Autonomous Scientists					
AI Scientist (v1)	4.5	6.0	35.5	10.5	43.5
AI Scientist (v2)	2.8	4.2	32.0	11.0	50.0
DeepResearcher	1.5	2.5	30.5	11.5	54.0
AI-Researcher	0.8	1.5	28.5	10.4	58.8

4.5 Phase III (Controlled Reporting): Integrity Error Rates

We audit controlled reports generated from the specified evidence bundle. Metric Exaggeration is the share of extracted claims that overstate performance relative to execution evidence; Ghost Citation is the share of references unresolved after bibliographic verification and adjudication. The denominators are claims and references, respectively, so the two rates characterize distinct error processes. Figure 4 shows Metric Exaggeration rates of 4.6–7.4% for writing specialists and 6.8–12.5% for E2E configurations. Ghost Citation rates span 4.8–5.2% and 8.1–14.8%, respectively. Bootstrap intervals quantify uncertainty, while audit records link each aggregate to absolute counts, supporting evidence, and adjudicated labels. The system ordering differs across the two error types, suggesting that numerical grounding and bibliographic validity capture complementary risks. Their persistence despite access to the evidence bundle motivates claim- and reference-level verification during report generation.

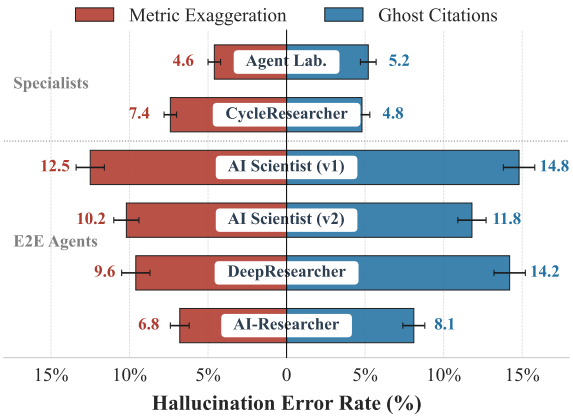

Figure 4: **Integrity Error Rates.** Left/right bars pool claims/references within each system. The embedded "Hallucination" axis denotes audit error rate; whiskers show bootstrap 95% CIs.

Answer to RQ3: Audited Integrity Errors

Averaged across the displayed systems under a common evidence interface, E2E configurations have higher rates for both audited error types than writing specialists.

5 Conclusion

AIS-Bench shows that evaluated AI Scientist configurations do not consistently preserve scientific fidelity across idea reconstruction, implementation, and reporting. In our reference-conditioned setting, the deficit is translating an intended mechanism into executable behavior: functional-logic deviations dominate classified implementation failures and can survive superficial execution checks. Controlled reporting exhibits metric exaggeration and citations unresolved after verification despite access to execution evidence. Thus, plausible ideas, runnable programs, and fluent reports are individually insufficient; reliable scientific agency requires traceable consistency among intended mechanisms, runtime behavior, evidence, and reported claims. AIS-Bench supports this diagnosis along published research trajectories, rather than unconstrained scientific novelty.

AI USE STATEMENT

Generative AI tools were used for manuscript drafting and language editing, literature retrieval and reference checking, and research execution. In the reported methodology, GPT-4o extracted structured idea tuples, assisted the construction of sanitized benchmark inputs, and supported evaluator cross-checking, while Claude 3.5 Sonnet scored specified subjective metrics. Thus, the benchmark contains AI-assisted synthetic inputs, but its source papers, repositories, and empirical results were not generated by AI. The authors manually verified references against primary sources and reviewed all benchmark artifacts, analyses, claims, and AI-assisted text. Generative AI was not used to prove mathematical claims. The authors determined the research questions and conclusions.

ETHICS STATEMENT

AIS-Bench uses public papers and repositories. Human annotations are limited to artifact-quality judgments and released only in anonymized form; no private or sensitive personal data are collected. Appendix G documents licensing, provenance, intended use, misuse risks, and release safeguards.

REPRODUCIBILITY STATEMENT

The appendices specify dataset construction, evaluation, and judge validation. The anonymous artifact provides sample-level outputs, configurations, prompts, scoring code, environment manifests, and execution logs.

REFERENCES

Anthropic. Claude 3.5 Sonnet. `https://www.anthropic.com/news/claude-3-5-sonnet`, 2024.

Jinheon Baek, Sujay Kumar Jauhar, Silviu Cucerzan, and Sung Ju Hwang. ResearchAgent: Iterative research idea generation over scientific literature with large language models. In *Proceedings of the 2025 conference of the nations of the Americas chapter of the association for computational linguistics: human language technologies (volume 1: long papers)*, pp. 6709–6738, 2025.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, et al. MLE-Bench: Evaluating machine learning agents on machine learning engineering. In *International Conference on Learning Representations*, volume 2025, pp. 50466–50494, 2025.

Guoxin Chen, Jie Chen, Lei Chen, Jiale Zhao, Fanzhe Meng, Wayne Xin Zhao, Ruihua Song, Cheng Chen, Ji-Rong Wen, and Kai Jia. Toward autonomous long-horizon engineering for ML research. *arXiv preprint arXiv:2604.13018*, 2026.

Hui Chen, Miao Xiong, Yujie Lu, Wei Han, Ailin Deng, Yufei He, Jiaying Wu, Yibo Li, Yue Liu, and Bryan Hooi. MLR-Bench: Evaluating AI agents on open-ended machine learning research. *Advances in Neural Information Processing Systems*, 38, 2025a.

Tingting Chen, Srinivas Anumasa, Beibei Lin, Vedant Shah, Anirudh Goyal, and Dianbo Liu. Auto-Bench: An automated benchmark for scientific discovery in LLMs. *arXiv preprint arXiv:2502.15224*, 13, 2025b.

Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, et al. ScienceAgentBench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *International Conference on Learning Representations*, volume 2025, pp. 96934–96990, 2025c.

Mingxuan Du, Benfeng Xu, Chiwei Zhu, Licheng Zhang, Xiaorui Wang, and Zhendong Mao. DeepResearch benchmark: A comprehensive benchmark for deep research agents. In *International Conference on Learning Representations*, volume 2026, pp. 42414–42448, 2026.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, et al. Towards an AI co-scientist. *arXiv preprint arXiv:2502.18864*, 2, 2025.

Sikun Guo, Amir Hassan Shariatmadari, Guangzhi Xiong, Albert Huang, Myles Kim, Corey M Williams, Stefan Bekiranov, and Aidong Zhang. IdeaBench: Benchmarking large language models for research idea generation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 5888–5899, 2025.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating language agents on machine learning experimentation. *arXiv preprint arXiv:2310.03302*, 2023.

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

Patrick Tser Jern Kon, Qiuyi Ding, Jiachen Liu, Xinyi Zhu, Jingjia Peng, Jiarong Xing, Yibo Huang, Yiming Qiu, Jayanth Srinivasa, Myungjin Lee, et al. EXP-Bench: Can AI conduct AI research experiments? In *International Conference on Learning Representations*, volume 2026, pp. 120555–120604, 2026.

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

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

Bodhisattwa Prasad Majumder, Harshit Surana, Dhruv Agarwal, Bhavana Dalvi Mishra, Abhijeetsingh Meena, Aryan Prakhar, Tirth Vora, Tushar Khot, Ashish Sabharwal, and Peter Clark. DiscoveryBench: Towards data-driven discovery with large language models. In *International Conference on Learning Representations*, volume 2025, pp. 4556–4579, 2025.

Yansheng Qiu, Haoquan Zhang, Zhaopan Xu, Ming Li, Diping Song, Zheng Wang, and Kaipeng Zhang. AI idea bench 2025: AI research idea generation benchmark. *arXiv preprint arXiv:2504.14191*, 2025.

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

Chenyang Shao, Yong Li, and Fengli Xu. SciNetBench: A relation-aware benchmark for scientific literature retrieval agents. *arXiv preprint arXiv:2601.03260*, 2026.

Yuning Shen, Julia E Borowski, Melissa A Hardy, Richmond Sarpong, Abigail G Doyle, and Tim Cernak. Automation and computer-assisted planning for chemical synthesis. *Nature Reviews Methods Primers*, 1(1):23, 2021.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, et al. PaperBench: Evaluating AI's ability to replicate AI research. *arXiv preprint arXiv:2504.01848*, 2025.

Jiabin Tang, Lianghao Xia, Zhonghang Li, and Chao Huang. AI-Researcher: Autonomous scientific innovation. *Advances in Neural Information Processing Systems*, 38:9481–9520, 2025.

Jiayu Wang, Weijiang Lv, Bowen Fu, Jing Fu, Jiayi Song, Lingyu Zhang, Lanxuan Xue, Luodi Chen, Zepeng Xin, Kaiyu Li, et al. Act as a real researcher: A suite of benchmarks evaluating frontier LLMs and agentic harnesses in research lifecycle. *arXiv preprint arXiv:2606.07462*, 2026.

Qingyun Wang, Doug Downey, Heng Ji, and Tom Hope. SciMON: Scientific inspiration machines optimized for novelty. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 279–299, 2024.

Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramaniam, Arjun R Loomba, Shichang Zhang, Yizhou Sun, and Wei Wang. SciBench: Evaluating college-level scientific problem-solving abilities of large language models. *arXiv preprint arXiv:2307.10635*, 2023.

Yixuan Weng, Minjun Zhu, Guangsheng Bao, Hongbo Zhang, Jindong Wang, Yue Zhang, and Linyi Yang. CycleResearcher: Improving automated research via automated review. In *International Conference on Learning Representations*, volume 2025, pp. 3669–3709, 2025.

Yixuan Weng, Minjun Zhu, Qiujie Xie, Qiyao Sun, Zhen Lin, Sifan Liu, and Yue Zhang. DeepScientist: Advancing frontier-pushing scientific findings progressively. In *International Conference on Learning Representations*, volume 2026, pp. 47981–48037, 2026.

Wanghan Xu, Shuo Li, Tianlin Ye, Qinglong Cao, Yixin Chen, Hengjian Gao, Yiheng Wang, Qi Li, Kun Li, Sheng Xu, et al. ResearchClawBench: A benchmark for end-to-end autonomous scientific research. *arXiv preprint arXiv:2606.07591*, 2026.

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

Yaping Zhang, Qixuan Zhang, Xingquan Zhang, Zhiyuan Chen, Wenwen Zhuang, Yupu Liang, Lu Xiang, Yang Zhao, Jiajun Zhang, Yu Zhou, et al. HiSciBench: A hierarchical multidisciplinary benchmark for scientific intelligence from reading to discovery. *arXiv preprint arXiv:2512.22899*, 2025.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. DeepResearcher: Scaling deep research via reinforcement learning in real-world environments. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 414–431, 2025.

A Sources and Coding Rules for the Benchmark Comparison

The sources for the benchmarks compared in Table 1 are ScienceAgentBench (Chen et al., 2025c), PaperBench (Starace et al., 2025), IdeaBench (Guo et al., 2025), AI Idea Bench 2025 (Qiu et al., 2025), SciNetBench (Shao et al., 2026), DeepResearch Bench (Du et al., 2026), Auto-Bench (Chen et al., 2025b), MLAgentBench (Huang et al., 2023), DiscoveryBench (Majumder et al., 2025), MLE-Bench (Chan et al., 2025), HiSciBench (Zhang et al., 2025), EXP-Bench (Kon et al., 2026), and MLR-Bench (Chen et al., 2025a).

Table 1 uses the following operational criteria. A checkmark indicates that the benchmark explicitly evaluates the corresponding output or capability: **Idea** covers hypothesis or method proposal; **Code** covers implementation or code-based scientific analysis; **Paper** covers a manuscript, literature review, or research report; **Dyn.** requires generated code execution or interaction with an executable environment; and **Dec.** requires at least two lifecycle-related stages to be scored separately with stage-specific inputs or rubrics. **Align.** requires Idea, Code, and Paper artifacts derived from the same published research trajectory. Merely providing a paper or codebase as task input does not satisfy the corresponding lifecycle column. The comparison records the task definitions in the cited benchmark versions; it is not a claim that unscored capabilities are absent from the underlying systems.

B Detailed Dataset Taxonomy and Statistics

This section distinguishes two dataset populations and two annotation levels. The **full corpus** contains 3,174 peer-reviewed machine-learning papers and is used to summarize corpus composition. The **executable core** contains 753 paper–repository pairs that pass the standardized execution filter and support construction of $\langle \mathcal{X}, \mathcal{I}, \mathcal{C}, \mathcal{P} \rangle$ quadruplets. Level 1 records the paper's primary contribution type. Level 2 assigns one operational diagnostic label, or *Other*, according to the benchmark interface for which the paper presents the clearest diagnostic challenge. Level-2 corpus counts are therefore not phase-specific evaluation sample sizes; executable-core subset sizes are reported separately with experimental records.

B.1 Level 1: Macro Task Typology

The Level-1 taxonomy uses the four mutually exclusive macro categories reported in Table 5: *Methodological Proposal*, *Theoretical Analysis*, *Benchmark/Dataset*, and *Empirical Discovery*. Methodological proposals constitute 87.93% of the collected corpus.

Table 5: **Level-1 article type distribution.** The full corpus contains 3,174 peer-reviewed machine-learning papers.

Article Type	Count	Ratio (%)
Methodological Proposal	2,791	87.93
Theoretical Analysis	221	6.96
Benchmark / Dataset	134	4.22
Empirical Discovery	28	0.88
Total	**3,174**	**100.0**

B.2 Level 2: Fine-Grained Diagnostic Categories

Level 2 is an operational stratification rather than an exhaustive description of a paper. Its labels group diagnostic challenges under Reference-Conditioned Idea Reconstruction (A1–A6), Code Implementation (B1–B7), and Controlled Reporting (C1–C4). Annotators assign both levels without access to evaluated system outputs, and an expert adjudicates disagreements. Table 6 gives the decision definitions and full-corpus counts. Execution failures observed after generation, including dependency, API, device, and package-version failures, belong to the Phase-II failure analysis and are not Level-2 corpus labels.

Table 6: **Level-2 diagnostic distribution.** Mutually exclusive operational labels over the 3,174-paper full corpus. Ratios use the full corpus as denominator; these counts are distinct from executable-core evaluation subset sizes.

ID	Operational Decision Definition	Count	Ratio
Phase I: Reference-Conditioned Idea Reconstruction			
A1	**Domain Transfer.** Adapting a method, representation, or insight across domains, modalities, or task settings.	20	0.6%
A2	**Efficiency Innovation.** Proposing efficiency-, resource-, or constraint-aware innovations such as low-rank adaptation, quantization, pruning, sparsity, or distillation.	20	0.6%
A3	**Mathematical Theory.** Formulating or analyzing ideas dominated by formal reasoning, derivations, bounds, kernels, information theory, or generalization arguments.	235	7.4%
A4	**Hybrid Architecture.** Combining multiple modules, paradigms, or representational pathways into one method, such as hybrid, retrieval-augmented, neuro-symbolic, or multimodal systems.	**660**	**20.8%**
A5	**Evaluation Methodology.** Designing benchmarks, evaluation protocols, metrics, robustness tests, fairness criteria, or adversarial evaluation settings.	336	10.6%
A6	**Problem Formulation.** Defining or reformulating the learning objective, task setup, constraints, curriculum, or loss structure.	29	0.9%
Subtotal		*1,300*	*41.0%*
Phase II: Code Implementation			
B1	**Tensor Operations.** Implementing tensor computation, reshaping, broadcasting, indexing, view/permute operations, or multi-dimensional tensor alignment.	30	0.9%
B2	**Gradient Computation.** Implementing gradient flow, custom autograd, differentiable relaxation, high-order gradients, GAN objectives, or optimization mechanics.	25	0.8%
B3	**State Management.** Managing persistent state across steps, modules, environments, caches, asynchronous calls, streaming inputs, or iterative trajectories.	364	11.5%
B4	**Graph Algorithms.** Implementing graph structures, sparse matrices, message passing, scatter/gather operations, adjacency handling, or graph indexing.	314	9.9%
B5	**Distributed Computing.** Implementing multi-device, multi-process, sharded, model-parallel, pipeline-parallel, CUDA/NCCL, or synchronization-dependent computation.	24	0.8%
B6	**Probabilistic.** Implementing probabilistic modeling, sampling, stochastic processes, latent variables, SDE solvers, MCMC, or random-seed-sensitive logic.	21	0.7%
B7	**Memory Optimization.** Implementing memory-efficient training, checkpointing, offloading, data loading, caching, mmap, or resource-aware execution.	**467**	**14.7%**
Subtotal		*1,245*	*39.2%*
Phase III: Controlled Reporting			
C1	**Result Analysis.** Interpreting empirical findings, ablations, scaling trends, sensitivity analyses, or empirical phenomena.	29	0.9%
C2	**Comparative Evaluation.** Comparing with baselines, prior work, leaderboards, ablation alternatives, or competing systems.	17	0.5%
C3	**Reproducibility.** Documenting algorithms, pseudo-code, notation, experimental settings, hyperparameters, random seeds, environments, code, data, or configurations.	106	3.3%
C4	**Ethical Considerations.** Reporting limitations, risks, fairness, privacy, bias, or misuse concerns grounded in the supplied problem context, method card, code, logs, and literature evidence.	**441**	**13.9%**
Subtotal		*593*	*18.7%*
Other / Unclassified		36	1.1%
Total		**3,174**	**100.0%**

B.3 Executable-Core Construction

The executable core is constructed from the full corpus through repository matching, environment resolution, and sandbox execution. A paper can contribute to the taxonomy even if its repository is unavailable or fails execution, but only samples that pass the filtering pipeline enter the executable core used for the main experiments. Table 7 summarizes the filtering process.

Table 7: **Executable-core filtering pipeline.** The full corpus is used for taxonomy analysis, while the executable core is used for downstream phase-wise evaluation.

Filtering Stage	Count
Raw peer-reviewed papers	3,174
Papers with public repositories	1,245
Environment-resolved repositories	864
Repositories passing the one-epoch check	753

Membership in the executable core requires standardized executability under the AIS-Bench sandbox, but it does not imply full reproduction of every claim in the original paper. Accordingly, AIS-Bench evaluates phase-aligned executability, implementation faithfulness, and evidence-grounded reporting, rather than complete claim-level reproduction of all original results.

C Judge Calibration and Evaluation Robustness

C.1 Annotation Protocol

Human validation serves two distinct purposes. First, annotators inspect a stratified set of 75 executable-core items to determine whether each distilled reference idea preserves the source paper's motivation, mechanism, and intended outcome. Second, held-out human labels evaluate the subjective scoring components: idea–reference alignment, code–idea consistency, and claim–evidence entailment. The held-out items are disjoint from examples used to calibrate prompts or score mappings. Alignment and consistency use ordinal labels; entailment uses the task labels defined in Appendix K. Independent labels are retained before adjudication so that agreement and adjudicated accuracy are not conflated.

Table 8: **Held-out human validation summary.** Reference-idea checks and judge–human comparisons use held-out annotations disjoint from prompt calibration. Correlations refer to the subjective metrics on their respective human scales.

Validation Item	Value
Reference-idea audit coverage	75 / 753 (10.0%)
Reference-idea inter-annotator agreement	$\kappa = 0.81$
Mean reference-idea faithfulness	4.31 / 5
Minimum reported judge–human correlation	$r > 0.88$

C.2 Judge Calibration

Judge prompts and score mappings are fixed on a calibration split before held-out evaluation. On a held-out item, the judge receives the same structured evidence available during benchmark scoring; the source manuscript is not added to the agent-facing interface. Agreement is computed separately for idea–reference alignment, code–idea consistency, and claim–evidence entailment. The summary threshold in Table 8 is the minimum reported Pearson correlation across these subjective metrics rather than a correlation obtained by pooling their labels.

C.3 Robustness and Consistency Analysis

Claude 3.5 Sonnet is the primary evaluator and GPT-4o is used for an alternative-judge analysis. Table 9 compares the system-level rankings induced by the two judges within each phase. This analysis concerns evaluator sensitivity; it does not validate objective execution metrics, which do not use an LLM judge.

Table 9: **Alternative-judge ranking agreement.** Kendall's τ compares system-level rankings produced by Claude 3.5 Sonnet and GPT-4o within each phase.

Phase	Inter-judge Kendall's τ
Idea Reconstruction	0.74
Code Implementation	0.71
Controlled Reporting	0.77

The correlations indicate substantial, but not perfect, evaluator agreement and therefore do not justify treating LLM-judged scores as evaluator invariant. Because generator and evaluator identities are not fully crossed in the native configurations, the analysis does not estimate a causal self-preference effect.

C.4 Run-to-Run Stability

To estimate variation from repeated proprietary-API calls, we score the same held-out items three times with unchanged prompts and inputs. Table 10 reports the standard deviation of each aggregate phase score across those three evaluator runs, in points on the 0–100 scale. These values quantify evaluator reruns, not variation over generated samples, training seeds, or benchmark resampling.

Table 10: **Run-to-run stability.** Standard deviations are computed over three runs on the 75-sample held-out test set.

Phase	Std. Dev.
Idea Reconstruction	1.7
Code Implementation	2.1
Controlled Reporting	1.3

C.5 Controlled Comparability and Metric Sensitivity

Phase assignments describe the interface evaluated, not an assertion that a system's native workflow is restricted to that role. SciMON and ResearchAgent are evaluated through the idea interface; DeepCode and CodeScientist through the code interface; and CycleResearcher and Agent Laboratory through the reporting interface. End-to-end systems are adapted to all three interfaces. Native profiles retain differences in models, tools, and scaffolding and must therefore be interpreted descriptively.

The three phase composites measure different constructs and are not compared as if their numerical scales were interchangeable. Cross-phase analysis instead uses matched quadruplets and a common human artifact-validity criterion applied separately to the three outputs. The pairing unit is the quadruplet; paired bootstrap intervals concern within-item phase differences under that common criterion, not differences among the native composites in Table 3. Matched-budget or matched-backbone analyses are sensitivity controls and do not, by themselves, identify a causal scaffold effect. Likewise, component-level and Strict-Pass@1-gated Code scores are reported as sensitivity analyses rather than alternative primary metrics.

C.6 Determinism and Reproducibility

All primary subjective evaluations use one recorded Claude 3.5 Sonnet configuration with temperature zero and unchanged prompts, input serialization, and scoring instructions. Temperature zero reduces sampling variation but does not guarantee bitwise determinism from a proprietary service. The exact API model identifier and invocation parameters are part of the evaluation configuration, and Table 10 reports the observed run-to-run variation.

D Interpretation of Qualitative Failure Labels

Phase-I feasibility failures require a concrete contradiction, such as an optimization path through a non-differentiable operation without an estimator. Phase-II shape failures require a runtime error attributable to incompatible tensor dimensions; a merely unusual shape is not sufficient. In Phase III, Metric Exaggeration denotes a result claim that overstates supplied execution evidence, whereas Ghost Citation denotes a reference unresolved after automatic lookup and manual adjudication. An unresolved reference is a verification failure, not by itself proof of fabrication or intent. These labels support item-level diagnosis and should not be interpreted as prevalence outside the evaluated outputs.

E Limitations

Reference conditioning and contamination. AIS-Bench measures recovery relative to a published research trajectory. Scientifically valid alternatives may differ from that trajectory and receive lower alignment or entity-overlap scores. The feasibility gate only checks for an evident violation

under its rubric and does not establish scientific validity. Masking and overlap checks reduce explicit solution cues, while post-cutoff subsets probe sensitivity to publication date. Because proprietary training corpora are undisclosed, none of these procedures can certify that an individual benchmark item was unseen during model training.

Cross-phase and cross-system comparability. Phase-specific composites measure different constructs, and their numerical scales alone do not establish relative task difficulty. Cross-phase statements are restricted to matched items scored with the common human criterion. Native configurations retain differences in models, tools, and scaffolding; even budget- or backbone-matched controls isolate only the factors explicitly held constant. The reported system differences should not be interpreted as causal effects of a particular scaffold.

Selection and execution validity. The executable core includes only papers whose public repositories can be resolved and run under the standardized sandbox, favoring maintained projects and conventional software stacks. One-epoch execution and 50-step synthetic trajectories test local executability and early optimization behavior, not full reproduction, real-data robustness, or claim-level scientific validity. The stage-decoupled protocol also omits feedback through which implementation results may revise an initial hypothesis.

Metric scope. AIS-Bench combines diagnostic proxies rather than treating any component as a complete measure of scientific quality. Entity overlap does not recognize all semantic equivalents; symbol recall does not establish behavioral equivalence; bibliographic existence does not establish relevance; and IMRaD distance measures requested-template compliance rather than writing quality. The composite scores inherit these limitations. Full scientific validity requires evidence beyond the benchmark, including domain-expert review.

Domain coverage. The corpus covers machine-learning papers from ICLR, ICML, and NeurIPS in 2023–2025. Its dominant methodological and computational traditions shape both the taxonomy and executable core. Results therefore characterize research agents within this setting; transfer to experimental sciences, different software ecosystems, and later publication periods remains an empirical question.

F Future Work

Exposure auditing. Broader temporal splits, model-specific cutoff analyses, and controlled paraphrases could better separate reconstruction from memorized surface form. Such analyses should report the exact similarity measure, threshold, and subset size.

Closed-loop evaluation. A subsequent benchmark could allow an agent to revise earlier artifacts after execution feedback. For example, after a tensor-shape failure, the agent could be asked to determine whether the defect is local to the implementation or follows from an inconsistent mechanism. This would evaluate revision behavior that the present stage-decoupled protocol does not measure.

Cross-domain and formal validation. Symbolic tools, proof assistants, and domain-specific validators could strengthen checks for claims that admit formal verification. Evaluations in bioinformatics, materials science, and other computational fields are needed before assuming that the present machine-learning taxonomy transfers to those domains.

G Responsible Release, Resources, and Broader Impacts

Data and code release. AIS-Bench is derived from publicly accessible papers and code repositories. Public accessibility does not imply an open-source license or permission to redistribute the original content. The submission artifact therefore contains derived benchmark metadata, interfaces, prompts, scoring code, and execution templates only where their inclusion is compatible with anonymity and third-party terms. A public release may provide source identifiers, retrieval scripts, and preprocessing instructions instead of copying restricted third-party material.

Compute resources. Benchmark construction uses isolated repository executions and API-based evaluation; AIS-Bench does not train a new foundation model. Phase-II dynamic checks use the resource limit specified in Appendix J, while subjective metrics use fixed prompts and temperature-zero evaluator calls. Repository-specific GPU requirements vary and are not summarized by a single accelerator configuration. Reproduction therefore requires the per-item environment manifest and execution record, rather than only an aggregate hardware description.

Licenses and existing assets. The source assets include peer-reviewed papers, public repositories, OpenReview metadata, Papers with Code links, and evaluated agent systems. Citations record scholarly provenance, but citation alone does not grant redistribution rights. Raw repository content is included in a release only after its license and terms permit redistribution; otherwise the benchmark stores provenance and reconstruction metadata without redistributing that content.

New asset documentation. The benchmark documentation specifies construction, inclusion criteria, provenance, interfaces, metric definitions, versioning, and known limitations. AIS-Bench is intended for diagnostic evaluation of research artifacts; it is not evidence that an agent can autonomously produce valid real-world discoveries, nor is its aggregate score a complete measure of scientific creativity.

Broader impacts. AIS-Bench may help identify failures in idea grounding, executable implementation, and evidence-faithful reporting before research agents are used in scientific workflows. Evaluation infrastructure can also accelerate systems that scale low-quality or misleading research outputs. The benchmark reduces, but cannot remove, this risk by rewarding traceable evidence and executable behavior. Its scores are not a substitute for expert scientific review.

Safeguards. AIS-Bench does not release a generative model. Release safeguards are license-aware redistribution, source provenance, documented intended uses, and exclusion of private or personally identifying information. Submission artifacts remain anonymized, and benchmark scores are explicitly not presented as proof of autonomous scientific validity.

H Sourcing and Construction Protocols

This appendix specifies the construction instructions and executable-core filters used for the **AIS-Bench** quadruplets. It documents the agent-facing transformations; passing these filters does not establish full reproduction of the source paper.

H.1 Automated Content Extraction

Idea distillation operates on the paper section identified as the methodology or proposed method. The protocol extracts a compact reference representation and deliberately excludes implementation details that belong to Phase II.

Protocol A: Scientific Idea Distillation

System Prompt. `You are a scientific content analyzer. Your goal is to decompose the methodology section of an input paper into a structured representation that captures the essence of the contribution.`

Input Specification.

- **Source:** Methodology or Proposed Method section of the research paper.
- **Target Output:** Structured JSON object.

Field Definitions.

- **`motivation`:** Identify the specific research gap or technical limitation addressed.
- **`mechanism`:** Summarize the core algorithmic novelty or structural change.
- **`outcome`:** State the primary performance metric or property targeted for improvement.

Constraints.

- **No implementation details:** exclude Python syntax, library references, and hardware specifications.

- **No hyperparameters:** exclude learning rates, batch sizes, and optimizer configurations.
- **Focus:** prioritize high-level algorithmic logic over engineering boilerplate.

Output. Return only a JSON object following the schema:

```
{"motivation": "...", "mechanism": "...", "outcome": "..."}.
```

H.2 Context Decontamination Protocol

To reduce explicit target leakage, we sanitize the paper's Introduction and cited prior-work context. The objective is to preserve the research problem while removing direct identifiers of the target method. This transformation does not remove information that a model may already have acquired during pretraining.

Protocol B: Context Sanitization via Named Entity Masking

System Prompt. `You are a data decontamination agent. Your task is to perform Named Entity Masking (NEM) to redact the identity of the proposed solution while maintaining the readability of the research problem.`

Task Instructions.

1. **Identify identity signals:** locate the proper noun, acronym, or unique name used by the authors to refer to their newly proposed method.
2. **Standardized redaction:** replace every occurrence of this name with the standardized mask token `[OUR_METHOD]`.
3. **Preserve context:** maintain general technical terminology and citations to prior work.
4. **Linguistic integrity:** preserve grammatical structure and academic flow.

Goal. The agent reading the output should understand the scientific challenge without directly identifying the proposed solution name.

After masking, exact method identifiers and solution-revealing spans are removed or rewritten before a second audit. Exact- and near-overlap checks identify residual textual cues; they are leakage diagnostics, not evidence about proprietary training data. Publication dates support post-cutoff sensitivity subsets only for model snapshots with documented knowledge cutoffs. Results on such subsets reduce, but cannot eliminate, uncertainty about prior model exposure.

H.3 Repository Validation and Filtering

The code component is derived from author-associated public repositories. The filter establishes local executability and a bounded evaluation footprint; it does not certify correctness, license status, or reproduction of every result.

Protocol C: Repository Filtering

Configuration Parameters.

- **Size constraint:** the cloned repository is capped at 500 MB.
- **Cloning policy:** each repository has a 300s timeout; code and configuration files are retained alongside the required documentation and dependency manifests.

Validation Criteria.

- **Manifest check:** the repository must document an entry point and provide sufficient dependency metadata for environment construction.
- **Dependency mapping:** use `pipreqs`-style import analysis to identify third-party packages while ignoring standard-library modules.
- **Execution sanity:** include only repositories that pass a one-epoch execution check in the sandbox.

I FULL EVALUATION PROTOCOLS FOR PHASE I (IDEA)

Phase I evaluates a structured hypothesis reconstructed from a sanitized target context. Structural Alignment measures agreement with the reference tuple; Constrained Novelty measures embedding distance to a leave-one-out retrieval corpus after a feasibility screen; and Reference Specificity measures reference-entity overlap. None of the three metrics alone establishes scientific novelty or validity. Figure 2 labels denote Reference-Conditioned Idea Reconstruction (Idea Generation), reference-entity overlap (Technical Specificity), Controlled Reporting (Paper Writing), IMRaD compliance (Structural Coherence), and cyclomatic similarity (maintainability); Pass@k uses $k = 1$.

I.1 IDEA-1: STRUCTURAL ALIGNMENT (S_{align})

This metric quantifies the semantic and technical alignment between the generated hypothesis $\hat{\mathcal{I}}$ and the reference idea $\mathcal{I}_{GT}$. The evaluation is performed component-wise across Motivation, Mechanism, and Outcome.

Scoring Rubric for Structural Alignment

Scoring Standard. Rate Motivation, Mechanism, and Outcome separately on a 1–5 scale:

- **5 (Equivalent):** the component preserves the reference gap, algorithmic mechanism, or intended outcome.
- **4 (Strongly aligned):** the component preserves the central meaning with minor technical differences.
- **3 (Moderately aligned):** the component shares the reference objective or technical family but differs in substantive details.
- **2 (Weakly aligned):** the component shares only broad domain context and misses the reference's specific content.
- **1 (Unrelated):** the component is absent, contradictory, or technically unrelated to its reference counterpart.

Calculation.

$$S_{align} = \frac{1}{3} \sum_{k \in \{\text{mot,mech,out}\}} \frac{r_k - 1}{4}, \qquad r_k \in \{1, \ldots, 5\}.$$

Thus, normalization is applied component-wise before averaging.

I.2 IDEA-2: CONSTRAINED NOVELTY (S_{nov})

This score is a retrieval-relative distance proxy. The feasibility gate removes credit when the judge identifies an explicit technical contradiction; it does not certify that a passing proposal is correct or publishably novel.

Protocol for Constrained Novelty and Feasibility

Novelty Calculation.

$$S_{nov} = \mathbb{I}_{feas}(\hat{\mathcal{I}}) \cdot \min_{d \in \mathcal{D}_{retrieval}} \text{clip}_{[0,1]}(1 - \cos(\mathbf{e}_{\hat{\mathcal{I}}}, \mathbf{e}_d)), \tag{9}$$

where $\mathbf{e}$ is produced by `text-embedding-3-large`. Generated ideas and retrieval records use the same serialization of Motivation, Mechanism, and Outcome and are ℓ_2-normalized before cosine similarity. For each sample, $\mathcal{D}_{retrieval}$ excludes the target paper, alternate versions of that paper, and duplicate records. The score depends on the coverage and snapshot of this retrieval corpus.

Feasibility Gate. The judge model performs a binary feasibility check:

- **Pass (1):** no explicit violation of the supplied mathematical or computational constraints is identified.
- **Fail (0):** the proposal contains a concrete contradiction, such as an impossible complexity claim or an optimization path requiring an unavailable derivative without an estimator.

I.3 IDEA-3: REFERENCE SPECIFICITY (S_{spec})

Reference specificity measures normalized technical-entity overlap with the distilled target idea. It is a reference-recovery metric, not a general measure of technical depth.

Protocol for Reference Specificity

Evaluation Logic.

1. **Entity extraction:** apply the same fixed technical-entity extraction and normalization procedure to $\hat{\mathcal{I}}$ and $\mathcal{I}_{GT}$; entity types include layers, objectives, and optimization methods.
2. **Overlap calculation:** for extracted entity sets $\mathcal{E}_{gen}$ and $\mathcal{E}_{GT}$, compute $S_{spec} = |\mathcal{E}_{gen} \cap \mathcal{E}_{GT}|/|\mathcal{E}_{gen} \cup \mathcal{E}_{GT}|$; assign zero when the union is empty.

Interpretation. High values indicate recovery of reference terminology. Semantically valid alternatives expressed through different entities can score lower, while lexical overlap alone does not establish mechanism equivalence.

I.4 PHASE I JUDGE SYSTEM PROMPT

The following system prompt is used to instruct the LLM judge during Phase I evaluation.

LLM-as-a-Judge: Phase I System Prompt

System Prompt. `You are an expert scientific reviewer for top-tier AI conferences. Your task is to evaluate a generated scientific hypothesis against a reference innovation distilled from a published paper.`

Task Instructions.

1. **Component alignment:** rate Motivation, Mechanism, and Outcome separately against their reference counterparts using the 1–5 rubric.
2. **Constraint check:** identify a concrete mathematical or computational contradiction; flag gradient-based mechanisms that require an unavailable derivative without a valid estimator.

Output Requirement. Return one concise justification and a JSON object with the schema `{"alignment": {"motivation": 1-5, "mechanism": 1-5, "outcome": 1-5}, "is_feasible": true|false}`. Reference Specificity is computed by the separate entity-overlap procedure and is not inferred from the judge's prose.

J ENGINEERING VERIFICATION DETAILS FOR PHASE II (CODE)

Phase II evaluates whether an agent can translate the supplied reference idea $\mathcal{I}_{GT}$ into executable code that preserves the requested mechanism. Providing the reference idea removes Phase-I reconstruction errors from this interface. The resulting metrics assess implementation evidence available to the benchmark; they do not prove full mathematical equivalence to the source repository.

J.1 CODE-1: CODE-IDEA ALIGNMENT (CIA)

The S_{CIA} metric uses an LLM judge to compare an AST-derived summary of the generated program with the reference mechanism. Because the judge does not observe every runtime state, this is a static semantic audit.

LLM-as-a-Judge: CIA System Prompt

System Prompt. `You are a senior software architect and ML researcher. Your task is to verify whether a generated Python implementation faithfully realizes a specific scientific hypothesis.`

Evaluation Protocol.

1. **AST inspection:** analyze the provided Abstract Syntax Tree (AST) summary, focusing on class definitions, inheritance, and core algorithmic loops.

2. **Logic verification:** check whether the unique mechanism, such as a specific attention mask or custom loss term, is explicitly implemented rather than merely described in comments.
3. **Consistency scoring:** assign a score from 1 to 5 based on technical fidelity to the reference idea.

Normalization. For judge rating $r_{CIA} \in \{1, \ldots, 5\}$, compute $S_{CIA} = (r_{CIA} - 1)/4 \in [0, 1]$ before phase-level aggregation.

Rubric.

- **5 (Faithful):** the inspected logic implements the reference mechanism and its required operations.
- **4 (Mostly faithful):** the mechanism is present, with a minor omission that does not replace its central operation.
- **3 (Generic):** the code uses a standard template that omits the requested innovation.
- **2 (Weak):** fragments resemble the requested mechanism, but required operations or data flow are absent.
- **1 (Invalid):** code logic contradicts the reference idea or core modules are missing.

J.2 CODE-2: FEATURE COMPLETENESS (S_{comp})

This metric uses static symbol-table analysis as a proxy for coverage of reference-defined functional blocks. It measures symbol recall, not functional equivalence.

Static Symbol Table Comparison

Algorithm.

1. **Reference extraction:** construct the critical set $\mathcal{F}_{GT}$ from function signatures and class methods in $\mathcal{C}^{GT}$ before inspecting generated code.
2. **Target extraction:** perform identical extraction on the agent's code $\hat{\mathcal{C}}$.
3. **Matching:** use one fixed normalization and fuzzy-name threshold for all systems, count each reference symbol at most once, and expose both settings with the scoring implementation.

Calculation.

$$S_{comp} = \frac{|\text{Matched Reference Symbols}|}{|\mathcal{F}_{GT}|}.$$

J.3 CODE-3: ENGINEERING QUALITY (S_{eng})

Engineering quality averages a single-attempt execution indicator and cyclomatic-complexity similarity: $S_{eng} = (\text{Pass@1} + S_{simp})/2$.

Protocol for Executability and Complexity

Strict Pass@1. One generated candidate is executed once under the item-specific harness. $\text{Pass@1} = 1$ only when that execution completes within budget with exit code 0; otherwise it is 0. Failures include:

- `RuntimeError`, such as tensor shape mismatch.
- `ImportError`, including unavailable or incompatible dependencies.
- Syntax errors, out-of-memory termination, and timeout.

Cyclomatic Similarity. We use the Radon library to compute McCabe's cyclomatic complexity $\mathcal{M}$:

$$S_{simp} = \exp(-|\mathcal{M}(\hat{\mathcal{C}}) - \mathcal{M}(\mathcal{C}^{GT})|).$$

Equal complexity can occur for behaviorally different programs, and a one-unit difference is penalized exponentially. Accordingly, S_{simp} is used only as a lightweight structural proxy and not as an independent measure of maintainability.

J.4 CODE-4: DYNAMIC FIDELITY ($S_{fidelity}$)

Dynamic Fidelity compares early loss trajectories under a shared, item-specific execution harness. It is evaluated only where the reference entry point can expose a scalar loss under standardized synthetic inputs.

Dynamic Loss Trajectory Alignment

Sandbox Environment.

- **Isolated execution:** Dockerized Ubuntu, 8 vCPUs, 32GB RAM.
- **Timeout:** 30 minutes per sample.

Fidelity Protocol.

1. Use the same harness, synthetic input specification, and fixed seed for $\hat{\mathcal{C}}$ and $\mathcal{C}^{GT}$.
2. Run $T = 50$ optimization steps and record one scalar loss per step.
3. Compare the paired loss vectors $\mathbf{L}_{gen}$ and $\mathbf{L}_{GT}$.

Calculation.

$$S_{fidelity} = \exp\left(-\frac{1}{T}\sum_{t=1}^{T}\frac{|\mathcal{L}_{gen}^{(t)} - \mathcal{L}_{GT}^{(t)}|}{|\mathcal{L}_{GT}^{(t)}| + \epsilon}\right).$$

Here $\epsilon > 0$ is fixed for all systems and prevents division by values near zero. High fidelity denotes pointwise agreement for this short synthetic trajectory; it does not imply agreement on real data, final convergence, or the source paper's reported results. A generated implementation that cannot produce the required trajectory receives $S_{fidelity} = 0$.

K INTEGRITY AUDIT PIPELINES FOR PHASE III (CONTROLLED REPORTING)

Phase III produces a controlled report from an evidence bundle $\mathcal{E}_{report}$ containing the sanitized problem context, distilled method card, reference implementation $\mathcal{C}^{GT}$, execution logs $\mathcal{L}_{exec}$, and a retrievable literature index. The source manuscript $\mathcal{P}$ is withheld and used only for offline provenance and artifact auditing. The three metrics below evaluate result-claim support, bibliographic resolvability, and compliance with the requested report structure. They do not evaluate every factual or rhetorical property of a scientific paper.

K.1 PAPER-1: RESULT-CLAIM SUPPORT (S_{fact})

This pipeline audits numerical and comparative result claims against the provided execution log. Its scope excludes general background statements and claims that cannot be represented by the supplied evidence.

Evidence–Claim Entailment Audit

System Prompt. `You are a scientific evidence auditor. Verify whether each numerical or comparative claim in the controlled report is supported by the supplied execution log. Do not use outside knowledge to fill missing evidence.`

Audit Protocol.

1. **Claim extraction:** split sentences containing metrics or comparative trends into atomic, independently checkable claims.
2. **Evidence mapping:** map each atomic claim to all required key–value records in the JSON execution log.
3. **Logic check:** determine the relationship between the `Log` as premise and the `Text` as hypothesis.

Relationship Definitions.

- **Entailment:** every numeric value is within the fixed 1% relative tolerance, or all logged values support the stated comparison.
- **Contradiction:** at least one mapped value or comparison conflicts with the log beyond tolerance. A contradiction is marked *Metric Exaggeration* only when it overstates performance.
- **Neutral:** the required experiment, baseline, or value is absent from the supplied log.

Output. Return atomic claims, entailment labels, exaggeration flags, and supporting log values.

Calculation.

$$S_{fact} = \begin{cases} \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \mathbb{I}[\text{NLI}(\mathcal{L}_{exec}, s) = \text{entailed}], & |\mathcal{S}| > 0, \\ 0, & |\mathcal{S}| = 0. \end{cases}$$

Here $\mathcal{S}$ contains the extracted auditable result claims. Neutral and contradicted claims are not counted as entailed. The zero-claim case receives zero rather than perfect credit, preventing omission from increasing the score.

For the error analysis in Figure 4, Metric Exaggeration is reported separately as

$$R_{\text{exag}} = \frac{\sum_{s \in \mathcal{S}} \mathbb{I}[s \text{ overstates logged performance}]}{|\mathcal{S}|}.$$

This rate is undefined for a report with no extracted claims; pooled rates use the total number of extracted claims across the displayed reports as the denominator.

K.2 PAPER-2: CITATION VALIDITY (S_{cite})

This protocol checks whether each generated reference can be resolved to an external bibliographic record. Bibliographic existence is distinct from topical relevance or evidential appropriateness.

Bibliographic Verification Protocol

Verification Logic.

1. **Parser:** extract title, authors, and year from the BibTeX or string-formatted reference.

2. **Database query:** perform fuzzy search via S2ORC, arXiv, and DBLP.
3. **Alignment check:** accept an automatic match when title similarity is > 0.9 and at least one parsed author overlaps. Manually adjudicate unmatched entries; retain an entry as unresolved only when no corresponding record is identified.

Calculation.

$$S_{cite} = \begin{cases} \frac{|\mathcal{R}_{resolved}|}{|\mathcal{R}_{gen}|}, & |\mathcal{R}_{gen}| > 0, \\ 0, & |\mathcal{R}_{gen}| = 0. \end{cases}$$

The zero-reference case receives zero to avoid rewarding omission. This score does not increase or decrease according to citation relevance.

The Ghost Citation analysis uses the complementary unresolved-reference rate

$$R_{\text{ghost}} = \frac{|\mathcal{R}_{gen} \setminus \mathcal{R}_{resolved}|}{|\mathcal{R}_{gen}|}.$$

For reports with no references, this error rate is undefined and contributes no reference to the pooled denominator. "Ghost Citation" therefore denotes a reference unresolved after automatic verification and adjudication; it does not assert intent or prove fabrication.

K.3 Paper-3: IMRaD Structural Compliance (S_{struc})

Structural compliance aligns the generated report with the IMRaD sequence requested by the benchmark. It is not a general coherence or writing-quality score.

IMRaD Template-Compliance Scoring

Algorithm.

1. **Move classification:** use one fixed discourse parser to map report spans to the move inventory used by the requested template, including Introduction/Background, Method, Results, and Discussion.
2. **Sequence alignment:** construct a sequence of rhetorical moves $\text{Seq}(\hat{\mathcal{P}})$.
3. **Distance calculation:** compute normalized Levenshtein distance from the requested IMRaD move sequence T_{IMRaD}.

Calculation.

$$S_{struc} = 1 - \frac{\text{Levenshtein}(\text{Seq}(\hat{\mathcal{P}}), \text{T}_{IMRaD})}{\max(|\text{Seq}(\hat{\mathcal{P}})|, |\text{T}_{IMRaD}|)}.$$

This score measures ordering and coverage of the requested rhetorical moves. It does not assess argument quality, stylistic clarity, or the validity of the reported science.

Phase-score aggregation. All component scores are normalized to $[0, 1]$ before aggregation. We use the equal-weight phase composites

$$\begin{aligned} S_{\text{Idea}} &= \tfrac{1}{3}(S_{align} + S_{nov} + S_{spec}), \\ S_{\text{Code}} &= \tfrac{1}{4}(S_{CIA} + S_{comp} + S_{eng} + S_{fidelity}), \\ S_{\text{Paper}} &= \tfrac{1}{3}(S_{fact} + S_{cite} + S_{struc}). \end{aligned}$$

For each diagnostic subset, we report the mean applicable phase score multiplied by 100. The phase averages in Table 3 are unweighted macro-averages over the four displayed subsets, not sample-frequency-weighted averages over the full corpus. The Code composite combines static and dynamic evidence, with $S_{eng} = (\text{Pass@1} + S_{simp})/2$. A non-executable output can receive static credit up to a theoretical upper bound of 62.5 on the 0–100 scale, since Pass@1 and dynamic fidelity are zero. Pass@1 separately records execution success. A Strict-Pass@1-gated Code composite and alternative component weights are sensitivity analyses; they are not substituted for the primary score. Because the three phase composites have different definitions, the common human criterion described in Appendix C, rather than these composites, is used for matched cross-phase analysis.