CREATIVEBENCH2.0: BENCHMARKING CREATIVE METHOD DISCOVERY AND FORESIGHT TOWARD RECURSIVE SELF-IMPROVEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Major advances in machine learning usually come from a small number of methods that work across tasks and scale well. So far, humans have found these methods. Recursive self-improvement asks whether AI systems can discover them on their own. Current evaluations cannot answer this question. They often cannot isolate gains in a target component from changes to the surrounding training and evaluation. They allow broad, unrestricted edits and often compare against reported numbers instead of reproduced baselines. We introduce **CreativeBench2.0**, a repository-level benchmark for measuring attributable method discovery. Each task is reverse-synthesized from a published research paper serving as an anchor, together with its associated code and evaluation setup. The synthesis process isolates one replaceable algorithmic component. A system may edit only this component. Its score is based on reproduced baselines and performance on both visible and hidden out-of-distribution settings. The benchmark has two tracks. *Discovery* requires systems to implement and improve a method. *Foresight* requires them to rank methods and identify methods that may fail under distribution shift, without running code. Across seven systems and 50 tasks, success is rare. Many attempts produce no valid experiment, and most valid attempts remain below the strongest baseline. Reference-context and interaction-budget comparisons show small observed gains. We also identify recurring failure patterns in agent trajectories. Foresight systems can rank methods better than chance, but they are poor at identifying distribution-sensitive methods. Execution, cross-setting improvement, and pre-execution judgment are separate challenges.

1 INTRODUCTION

Major advances in machine learning often come from a small number of method changes that transfer across settings and scale well (He et al., 2016; Kingma & Ba, 2015; Vaswani et al., 2017). So far, these changes have been discovered by human researchers. If AI systems could discover them independently (Fawzi et al., 2022; Romera-Paredes et al., 2024; Novikov et al., 2025), AI progress would no longer be limited by the number of researchers, which is the premise of recursive self-improvement (Lu et al., 2024; Yamada et al., 2025). However, this ability is not yet well measured. An agent may improve a metric through an algorithmic idea, or through tuning, extra capacity, or engineering, and current evaluations cannot reliably distinguish these causes.

Recent language-model agents perform well on code generation (Chen et al., 2021; Jain et al., 2025), software engineering (Jimenez et al., 2024; Yang et al., 2024), data analysis (Lai et al., 2023; Chen et al., 2025b), and ML engineering (Huang et al., 2024; Chan et al., 2025). These results show that agents can execute complex workflows when the objective and intervention target are already specified. Method discovery begins one step earlier: the agent must decide what should be changed and show that the resulting gain comes from that change. This leads to a more fundamental question: *Can an AI agent propose generalizable, scalable, and attributable machine learning methods?*

Answering this question requires more than optimizing a given metric. Given data, scientific context, and strong public baselines, an agent must understand existing methods, identify a component worth improving, propose a change, and test whether its gains persist across datasets, model scales, or environments. This is an instance of open-ended discovery, where the agent chooses what to

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.

investigate rather than only executing a supplied query (Agarwal et al., 2025; Wang et al., 2026b). It must also validate its proposal against accumulated evidence (Weng et al., 2026; Chen et al., 2026).

Method discovery therefore involves two forms of research ability. The first is empirical: implementing and testing a proposed change. The second is predictive: deciding which methods are promising and which may fail under distribution shift before spending the experimental budget. We study both abilities under a controlled definition of method discovery: identifying an imperfect algorithmic component and proposing a change that remains competitive with strong reproduced baselines across visible and hidden settings. Existing research-agent benchmarks face three main challenges. First, they cannot reliably attribute gains to a target method when capacity, training, and surrounding code may also change. Second, end-to-end science benchmarks often cannot distinguish strong from weak methods because their long evaluation chains make method-level contributions difficult to isolate (Chen et al., 2025b). Third, existing algorithm-discovery systems are often built around one fixed domain, component, or search space, such as a CUDA kernel, a post-training loss, or a tensor program (Fawzi et al., 2022; Romera-Paredes et al., 2024; Ma et al., 2024; Novikov et al., 2025). A general benchmark should instead cover many scientifically meaningful components while keeping each individual task controlled. These components should be grounded in published research, supported by competing methods, and shown through reproduced baselines to have effects across settings.

We introduce **CreativeBench2.0** (Figure 1), a benchmark built around the following question: Can an agent improve a scientifically meaningful algorithmic component under a fixed training protocol and restricted edit scope, pass a calibrated success threshold, and remain competitive with the strongest baseline on hidden out-of-distribution (OOD) settings? The benchmark separates the two stages of method discovery. **Discovery** evaluates whether an agent can propose, implement, and experimentally refine a method. **Foresight** evaluates whether it can rank candidate methods, explain their mechanisms, and identify OOD-sensitive methods before execution. This separation distinguishes experimental improvement from pre-execution scientific judgment. We enforce attribution through a fixed evaluation, a fixed training protocol, and one restricted editable component. Generalization is measured on visible and hidden OOD settings, with scores calibrated against reproduced public baselines and aggregated by a geometric mean. To construct diverse tasks at scale, we introduce a **repo-grounded multi-agent synthesis pipeline**. Each task starts from a published paper and traces its contribution to versioned research code. The pipeline selects a component only when it has a clear interface, multiple competing published implementations, and reproduced baselines that can distinguish their behavior. It then freezes the surrounding training and evaluation flow. This produces the shared-backbone and swappable-method structure needed for attributable discovery while expanding coverage across repositories and research domains.

We evaluate frontier systems on both tracks. Reliable method discovery remains difficult: many attempts fail to produce valid experiments, and valid improvements often fail to generalize across settings. We also observe recurring failure patterns in agent trajectories. Context and Iterative performance show small observed differences. Models can rank methods before execution, but are much less reliable at predicting OOD robustness. These results identify valid execution, generalizable method improvement, and robustness foresight as distinct bottlenecks for current AI research agents.

Our three contributions are:

- An open-ended multi-agent pipeline. We develop a pipeline that identifies scientifically meaningful components and constructs auditable tasks from published research artifacts.
- CreativeBench2.0, a benchmark for machine learning method discovery. We evaluate method discovery under a fixed backbone, restricted edit scope, reproduced baselines, and visible and hidden OOD settings.
- A systematic evaluation of Discovery and Foresight. We show that current systems struggle with reliable method discovery and OOD prediction.

2 METHOD

2.1 OVERVIEW

We measure machine learning method discovery. Each task makes its strong baselines public and fixes the training and evaluation protocols. Under these conditions, can an agent propose and implement

Table 1: CreativeBench2.0 compared with representative agent benchmarks. ✓ denotes full support, ○ partial or indirect support, and ✗ absence. **Attrib. scope** (attributable scope) restricts gains to a single editable algorithmic component; **Baseline-calibrated** scores against reproduced method baselines; **Hidden OOD** uses held-out distribution-shifted settings; **Foresight** evaluates pre-execution method ranking and OOD-sensitivity; **Auto. synth.** denotes automated task construction. Entries summarize our reading of each paper's released protocol.

	What the agent does	Measurement design						Coverage
Benchmark	**Objective**	**Discovery**	**Attrib. scope**	**Baseline-calibrated**	**Hidden OOD**	**Foresight**	**Auto. synth.**	**Scale**
Execution- and engineering-oriented benchmarks								
SWE-bench (Jimenez et al., 2024)	Resolve repo issues	✗	✗	✗	✗	✗	✓	2,294 issues, 12 repos
MLAgentBench (Huang et al., 2024)	Improve an ML script	○	✗	○	✗	✗	✗	13 tasks, 5 categories
MLE-bench (Chan et al., 2025)	Win Kaggle medals	✗	✗	○	✗	✗	✗	75 Kaggle competitions
KernelBench (Ouyang et al., 2025)	Speed up GPU kernels	✗	○	○	✗	✗	✗	250 kernels, 3 levels
ALE-Bench (Imajuku et al., 2025)	Optimize solvers	○	✗	○	○	✗	✗	40 contest problems
Research-workflow and replication benchmarks								
ScienceAgentBench (Chen et al., 2025b)	Write workflow code	✗	✗	○	✗	✗	✗	102 tasks, 4 fields
PaperBench (Starace et al., 2025)	Replicate a paper	✗	✗	✗	✗	✗	✗	20 ICML papers
MLR-Bench (Chen et al., 2025a)	Run open-ended workflow	○	✗	✗	✗	○	✗	201 tasks, 9 topics
Method-level improvement benchmarks								
PostTrainBench (Rank et al., 2026)	Post-train a model	✗	✗	○	✗	✗	✗	28 model–bench. pairs
AIRS-Bench (Lupidi et al., 2026)	Beat published SOTA	○	✗	○	✗	✗	○	20 tasks, 7 categories
FIRE-Bench (Wang et al., 2026a)	Rediscover findings	✗	✗	✗	✗	✗	○	40 tasks, 40 papers
CreativeBench2.0 (ours)	**Discover a method**	✓	✓	✓	✓	✓	✓	**50 tasks, 8 ML domains**

an attributable change to one component? Such a change must be simple, must be effective across visible and hidden settings, and must stay competitive with the strongest verified baselines that the task includes. CreativeBench2.0 addresses this ability with two parts, which we present in turn. **(A) A benchmark design** (§2.2) defines a task as "fixed backbone + restricted editable scope + visible/hidden settings" and gives it a score calibrated against baselines. **(B) An open-ended discovery synthesis pipeline** (§2.3) reverse-synthesizes such tasks from research code and from artifacts grounded in published literature. The two parts support each other, and both share one belief: attributability comes from controlling upstream inputs (a fixed evaluation, a fixed training protocol, a single editable component).

2.2 Benchmark Design

Task setup and controlled evaluation. Each task provides a frozen research codebase, an evaluation protocol, and one editable algorithmic component. All systems use the same interface and isolated per-task context. Outside the editable component, the training script, optimizer, duration, data processing, evaluator, metrics, seeds, and time limit remain fixed. Task-specific compute and parameter caps further limit resource changes. The evaluator inserts each submission into the fixed backbone, localizing implementation differences to the component. Each task also includes at least three public baselines with distinct mechanisms, implementations, and literature or source evidence. These are verified competitors rather than an exhaustive list of all methods. Visible settings support experimentation, while at least one hidden OOD setting tests transfer across data, model scale, noise, or environment dynamics. Agents receive only visible feedback during interaction, and final evaluation covers all required settings.

Evaluation protocol. The benchmark has two tracks. **Discovery** requires an agent to implement and refine the editable component using the task description, code, baselines, and research context under a fixed interaction budget. Reference information and interaction opportunities may vary, while the task interface and evaluator remain fixed. **Foresight** presents the same research problem and candidate methods but forbids code execution. Agents must rank the methods, explain their mechanisms, and optionally flag OOD-sensitive methods. Ranking, sensitivity prediction, and explanation quality are scored separately and combined into a composite score.

For Discovery, we freeze one scoring specification. It records whether a higher or a lower metric value is better, and it maps each objective to $[0, 1]$. For each objective term, the weakest baseline defines the floor, and the strongest baseline serves as a calibration reference that we aim to map to

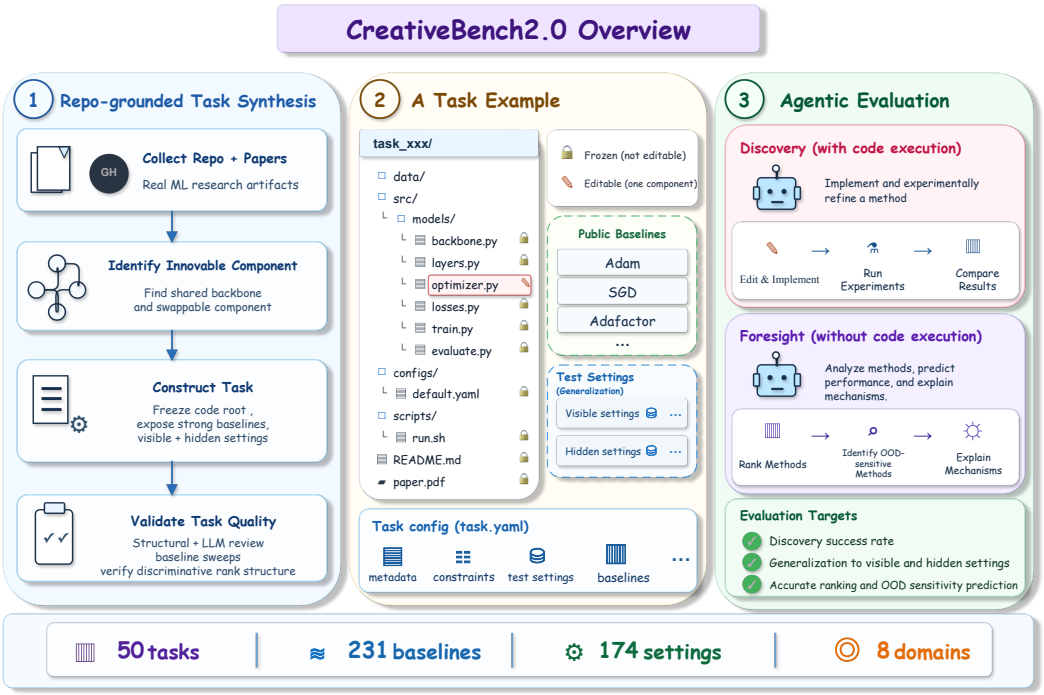


Figure 1: Overview of **CreativeBench2.0**. A repo-grounded synthesis pipeline constructs attributable method-discovery tasks from real ML research artifacts by identifying a shared backbone and a swappable algorithmic component, while freezing the surrounding training and evaluation pipeline. Each task exposes strong public baselines and visible/hidden settings. **Discovery** evaluates method implementation and experimental improvement, whereas **Foresight** evaluates pre-execution method ranking, OOD-sensitivity prediction, and mechanism understanding.

0.5. This holds except in edge cases where the score is bounded. We then weight the terms, apply the constraint penalties, and combine them with a geometric mean over all required visible and hidden settings. The result is the Discovery score D_t. Invalid, incomplete, and all-zero records receive zero.

We define Discovery success by the fixed threshold $D_t > 0.5$. We also compare D_t with B_t, the strongest included baseline's aggregate score, and report strict wins when $D_t > B_t$. Because we calibrate each term separately, aggregate the terms, and handle edge cases where the score is bounded, B_t need not equal 0.5. Passing the threshold and strictly winning against the baseline are two distinct performance criteria. Neither criterion establishes novelty. We have completed a qualitative human review. It supports the results we inspected, and it identifies substantive changes of mechanism in some submissions (Appendix C.3). The exact scoring rules appear in Appendix A.2.

Auditable records. Each attempt retains its results, transcript, test log, condition configuration, submission status, and submitted code or documented absence. Tables and figures are regenerated from these frozen records, linking each reported outcome to the experiment and component.

2.3 Task Synthesis Pipeline

Repo- and paper-grounded task discovery. Each task starts from a research repository that implements several methods in one shared pipeline. Survey reads the repository without installing or running it. It looks for four things: an official evaluation script, at least three methods that work in different ways, one component that can be isolated, and an environment we can build. For repositories that pass, the pipeline picks the component that agents will edit and checks that published papers compete on it. The code tells us the interface of that component and how each method implements it. The papers tell us where each method came from, which settings it was tested in, and how the methods ranked. From the source we copy only what is needed to reproduce the setup: the repository

URL, the commit, and the environment. Everything an agent reads — the task description, the research question, the edit scope, the background, and the baseline set — is written and checked by our pipeline.

Multi-agent construction. Agents build each task in stages, and an orchestrator decides when a stage is done (Table 18). Scout and Verify fix the component and require at least three published methods that change that same component. Each method must have a real citation and code we can reach. Analyze then clones the repository and runs the official evaluation. It first reproduces the published numbers, then copies the methods into our task layout and checks that the baselines still finish in the same order. Scope marks the lines an agent may edit. Baseline writes each competing method inside that edit scope. Assemble packages the runnable task with its visible and hidden settings. Validate and Review check the task mechanically, test the parser, and ask an LLM judge to score its quality. If a stage fails in a way we can fix, that stage runs again with the failure as feedback. If a source has too few papers, too little method diversity, or no usable evaluation, we drop it and look for another component or repository. Review also writes a checklist for a human reader (Appendix D.1). Stage contracts and retry rules are in Appendix D.2.

Evaluation fidelity and task admission. When a task comes from an upstream repository, we reuse an evaluation script that already exists there. We keep its metric, its data or official simulator, its model scale, and its main settings. Official simulators are allowed, but their settings must follow the source experiments. We never tune a setting until the ranking looks the way we want.

We then run every baseline ourselves, in every required visible and hidden setting, under the seeds the task declares. This tells us whether the evaluation can actually tell the methods apart. A task that merely runs is not enough: the edit scope, the settings, and the measured baseline numbers must all be valid. We reject a task if measurements are missing, if the evaluation is invalid, or if every baseline falls to noise. We keep a task when its baselines finish in a clear order. We also keep a task whose overall order has ties, as long as every required setting is informative and the ties come from a stable order change between settings, most often a hidden/OOD reversal. Stage gates are in Appendix D.2, and the rules for accepting or discarding a task after the run are in Appendix D.3.

Construction evidence. In the frozen final set, all 50 tasks keep their task package, their raw baseline runs, and their rank report. Of these, 37 have baselines that finish in a fully separated order. The other 13 keep a tie in the overall order, and every tie comes from methods that change places between settings; we keep these tasks because such crossings test generalization directly. Separately, 25 tasks contain at least one pair of methods whose order flips between the overall and the hidden setting. Baselines also carry their sources with them: 227 of the 231 baseline files include a structured paper reference, and 153 of these give an arXiv identifier. Appendix D.1 reports the Survey and code-origin audits.

3 Experiments

Experimental setup. We evaluate seven systems on 50 tasks that pass the construction and baseline-ranking gates in §2.3. Together, these tasks include 231 public baselines and 174 settings, with 102 visible and 72 hidden. All systems use the same frozen task packages and isolated per-task contexts. We fix the code roots, editable scopes, commands, metrics, seeds, and time limits. Under the primary Reference/Iterative protocol, each model makes one Discovery attempt per task, with up to 20 tool actions and three official tests. Runtime feedback exposes only visible-setting metrics. The submitted candidate is evaluated on all required visible and hidden settings. Appendix A describes the protocol and task inventory. RQ1 measures Discovery performance with reference context under the *Iterative* protocol (§3.1). RQ2 tests the effects of reference context and additional experimentation through two paired comparisons (§3.2). RQ3 evaluates method ranking, OOD-sensitivity prediction, and mechanism explanation without code execution (§3.3).

3.1 RQ1: How Often Do Agents Succeed at Discovery?

Under the frozen Discovery criterion, 14.6% of the 350 attempts succeed (Figure 2). A run succeeds only if it produces a complete, scoreable experiment with $D_t > 0.5$. This fixed threshold uses

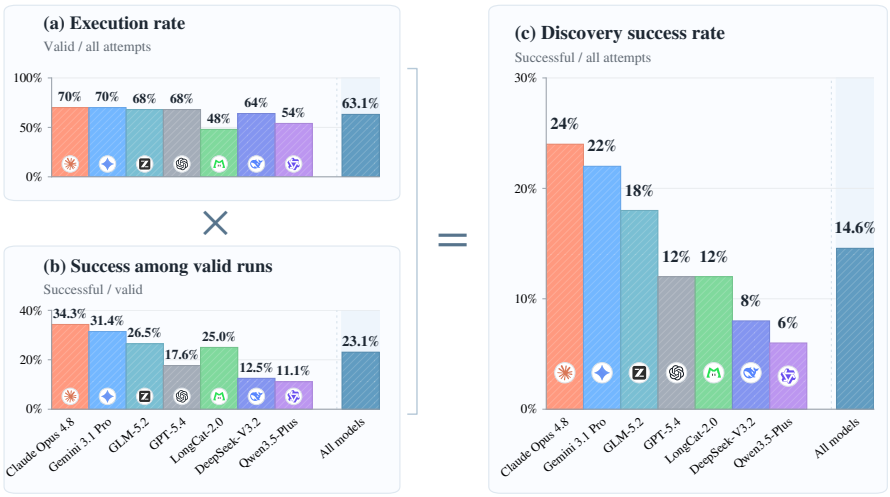


Figure 2: **Discovery success has two stages.** A run must first produce a valid experiment and then exceed the success threshold.

term-wise baseline calibration, and geometric aggregation rewards methods that perform well across settings (§2.2). Claude has the highest observed Success Rate. Comparing directly with the task's strongest aggregate baseline gives a 14.9% observed strict-win rate, but the two criteria select different submissions (Appendix Table 5). A strict win requires only a positive observed margin; it does not require statistical significance. We also check seeds for a fixed candidate and compute bootstrap intervals over tasks (Appendix B.1). The decomposition in Figure 2 shows two main obstacles: producing a scoreable experiment and improving the method enough to pass the threshold. Most scoreable methods still fall short. Similar Execution Rates can also produce different Discovery Success Rates, as illustrated by Claude and GPT. Execution reliability therefore explains only part of the performance gap (Huang et al., 2024; Chan et al., 2025; Jimenez et al., 2024).

Finding #1

Discovery success depends on both execution and method quality. Only 14.6% of attempts succeed, and Claude has the highest observed Discovery Success Rate.

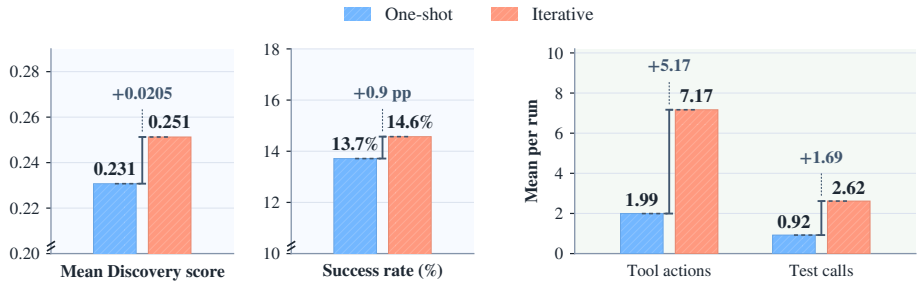
3.2 RQ2: Do Reference Context and More Experimentation Help Discovery?

RQ2 asks whether reference context and additional experimentation improve method discovery. We compare two interventions with the frozen runs from RQ1. First, we remove the curated reference context while keeping 20 tool actions and three tests. Second, we compare Iterative with *One-shot*. One-shot allows two actions and one test, with no revision after feedback. Iterative allows revision and retesting with visible feedback. Both comparisons use 350 matched model–task pairs from seven models and 50 tasks. We compare paired changes in D_t, with Execution Rate and hidden-setting scores as complementary measures. The reference package provides research background, prior methods, source links, comparison guidance, and visible baseline scores. The pooled Reference score difference is $+0.0217$, with an interval spanning zero (Table 2). Runtime dependency differences prevent isolating the effect of reference context; Appendix B.2 reports the sensitivity analysis.

Iterative has a small observed average gain over One-shot, while using 3.6 times as many tool actions (Figure 3). The gain also hides substantial turnover: some One-shot successes are lost while new Iterative successes appear. Most pairs with different outcomes produce scoreable submissions in both conditions, so the difference is not explained only by execution failures. The two conditions differ at the same time in budget, in feedback, in revision, and in the opportunity to retain a candidate. The campaigns also differ in runtime, and GLM differs in its thinking configuration; these further differences prevent us from isolating the effect of additional experimentation (Appendix B.3).

Table 2: **Paired reference-context results across the full task matrix.** Means and paired changes are computed on the same tasks across 350 matched pairs from seven models and 50 tasks. Δ denotes Reference minus No ref.; Claude shows the largest gain in both scores. Details appear in Appendix B.2.

	Discovery score			Hidden score		
Model	No ref.	Reference	Δ	No ref.	Reference	Δ
GPT-5.4	0.238	0.278	+0.040	0.257	0.291	+0.034
Claude Opus 4.8	0.254	0.343	**+0.089**	0.265	0.352	**+0.087**
Gemini 3.1 Pro	0.286	0.296	+0.010	0.306	0.334	+0.028
GLM-5.2	0.272	0.290	+0.019	0.293	0.301	+0.007
DeepSeek-V3.2	0.186	0.203	+0.017	0.193	0.206	+0.013
Qwen3.5-Plus	0.181	0.171	−0.011	0.197	0.185	−0.012
LongCat-2.0	0.190	0.177	−0.012	0.191	0.209	+0.018
Pooled	**0.230**	**0.251**	**+0.022**	**0.243**	**0.268**	**+0.025**



(a) Discovery performance **(b) Interaction cost**

Figure 3: **Observed One-shot–Iterative differences.** (a) Mean Discovery Score and Success Rate. (b) Mean tool actions and test calls. Blue denotes One-shot and orange denotes Iterative. Reference context is fixed across all 350 pairs. Test calls count as tool actions.

Finding #2

Similar aggregate success rates mask substantial turnover: 29 model–task pairs succeed only under Iterative and 26 only under One-shot. Iterative uses 3.6× as many tool actions, with three more observed successes across 350 pairs.

3.3 RQ3: Can Models Rank and Explain Methods Without Running Them?

Each system receives a research brief and a description card for each candidate method. Without executing code, it ranks the methods, explains their mechanisms, and may flag methods that are sensitive to distribution shift. We measure pairwise ranking accuracy using the frozen baseline sweeps and use F1 to evaluate the detection of ranking reversals under hidden shifts.

As shown in Figure 4, models rank methods above chance, and their predictions broadly match the hidden-only rankings. However, they often miss individual reversals, with Balanced Accuracy remaining close to chance (Appendix Table 12). Models provide most of the requested mechanism explanations. We report the GPT-5.4-rated explanation quality E_t and the composite Foresight score F_t as descriptive diagnostics. Deterministic ranking and flagging metrics remain the primary cross-model measures (Appendix B.4, Table 13).

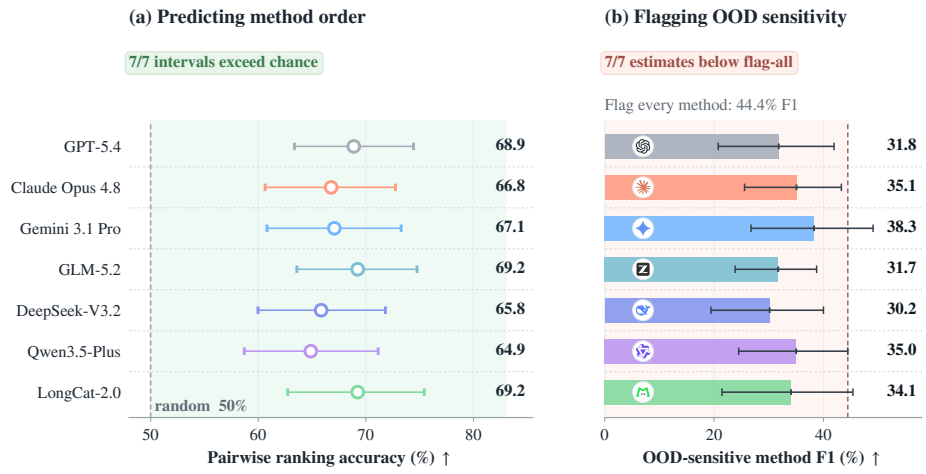


Figure 4: **Models recover broad method order but struggle with hidden failures.** (a) Pairwise ranking accuracy is above chance for all models. (b) F1 for identifying methods that are sensitive to distribution shift is below the flag-all baseline.

Finding #3

Models recover the broad method order but struggle to identify which methods fail under distribution shift.

3.4 Where Discovery Runs Fail

Across the Iterative cohort, runs that exhaust the test budget or explore at least three distinct code states have lower observed success than their complementary groups (Figure 12). No trajectory that spends all three calls on an unchanged state succeeds. Earlier benchmarks also diagnose agents at the level of individual trajectories (Huang et al., 2024; Chan et al., 2025; Chen et al., 2025b). Following them, we treat these patterns as descriptive contrasts only, because the groups differ in which tasks and models they contain, in difficulty, and in the rules for stopping a run. Full profiles and additional cases appear in Appendix C.

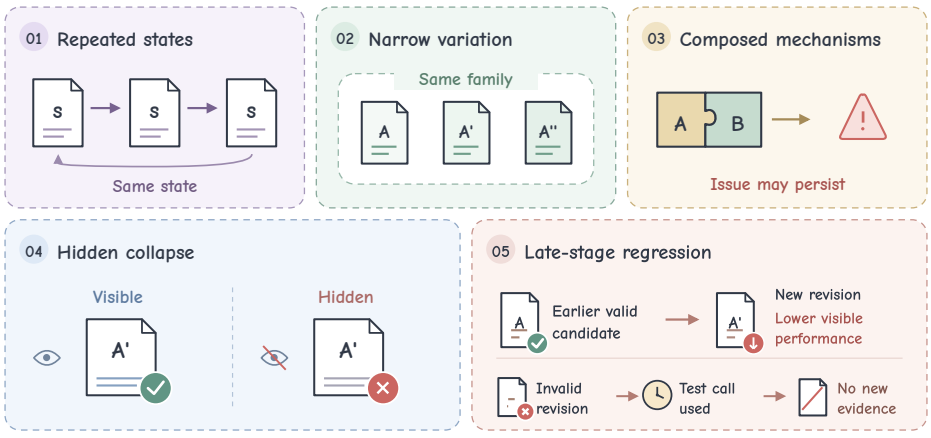


Figure 5: **Observed failure patterns in discovery runs.**

Figure 5 separates five failure patterns. *Repeated states* and static rejections can consume tests without new performance evidence. *Narrow variation* explores one method family, so more code

states need not represent more distinct ideas. *Composed mechanisms* may add complexity without resolving the original failure mode; combining components does not ensure that they complement one another. *Hidden collapse* occurs when visible gains fail to transfer to hidden settings, which agents cannot inspect during testing. *Late-stage regression* occurs when a new revision performs worse than an earlier valid candidate. A method switch may explore a new idea but lower performance on several visible settings, while invalid revisions can consume test calls without producing new evidence.

4 Related Work

Scientific discovery. Recent work uses language-model agents as scientific researchers. The AI Scientist and AI Scientist-v2 automate the process from idea generation to experiments and papers (Lu et al., 2024; Yamada et al., 2025). Other systems study open-ended hypothesis generation, frontier discovery, and evidence-based claims (Agarwal et al., 2025; Weng et al., 2026; Chen et al., 2026). ScienceAgentBench and FIRE-Bench evaluate executable scientific workflows and the rediscovery of scientific findings (Chen et al., 2025b; Wang et al., 2026a). These works measure discovery or research productivity. Tool-augmented systems connect scientific agents to domain tools and laboratory execution for chemical discovery (M. Bran et al., 2024; Boiko et al., 2023; Xu et al., 2025). Our work asks a question: can agent improve one algorithmic component over reproduced baselines?

Automated research and research benchmarks. SWE-bench, MLAgentBench, and MLE-bench evaluate software execution, experimentation, and ML engineering. RE-Bench and MLS-Bench extend this to frontier research and AI development (Jimenez et al., 2024; Huang et al., 2024; Chan et al., 2025; Wijk et al., 2025; Lyu et al., 2026). PaperBench and MLR-Bench test paper replication and open-ended research, while KernelBench and ALE-Bench focus on systems and algorithm engineering (Starace et al., 2025; Chen et al., 2025a; Ouyang et al., 2025; Imajuku et al., 2025). PostTrainBench and AIRS-Bench study post-training and literature-grounded research (Rank et al., 2026; Lupidi et al., 2026). Most of these benchmarks allow broad edits and rely on leaderboards, rubrics, or judges. They therefore cannot clearly attribute gains to one algorithmic idea. CreativeBench2.0 fixes the code and evaluation, calibrates scores with reproduced baselines, and tests methods on held-out settings.

Self-evolving agents. Self-evolution methods improve programs, workflows, or agents through search and feedback. FunSearch, Eureka, AlphaTensor, and AlphaEvolve use executable evaluation to find better programs or algorithms (Romera-Paredes et al., 2024; Ma et al., 2024; Fawzi et al., 2022; Novikov et al., 2025; Zhou et al., 2025). Self-Refine and Reflexion improve outputs through feedback, while the Darwin Godel Machine also modifies the agent code (Madaan et al., 2023; Shinn et al., 2023; Zhang et al., 2026). TaskCraft, SWE-smith, and MLE-Smith scale the supply of tasks or training artifacts (Shi et al., 2026; Yang et al., 2025; Qiang et al., 2026). These systems provide useful improvement mechanisms but do not directly test method-level creativity under controlled comparison. CreativeBench2.0 combines constrained method editing with reproducible baselines and cross-seed validation, testing whether an agent's idea is both effective and robust.

5 Conclusion

We introduced CreativeBench2.0, together with a multi-agent pipeline that reverse-synthesizes tasks from real research artifacts. The benchmark fixes the code and evaluation protocol, restricts edits to a single algorithmic component, and evaluates improvements against public baselines in visible and hidden settings. It contains 50 tasks across eight domains and 231 baselines, covering experiment-driven Discovery and execution-free Foresight. Our results show that reliable method discovery remains difficult. The Discovery Success Rate is only 14.6%, compared with an Execution Rate of 63.1%, indicating that running a valid experiment does not imply an effective method. Trajectory analysis also reveals recurring failure patterns, including ineffective search, repeated measurement, and changes without substantial mechanistic improvement. Additional reference information and interaction budgets provide only limited gains (Appendix B.2). In Foresight, models rank methods better than chance before execution but remain weak at predicting robustness under OOD settings. Overall, CreativeBench2.0 separates three challenges for AI research agents: executing valid experiments, proposing competitive methods that generalize across settings, and predicting performance under hidden distribution shifts.

AI USE STATEMENT

For manuscript preparation, generative AI tools assisted with editing for readability and with LaTeX formatting and template compliance. In the reported research, models were invoked through APIs as evaluated agents and as components of task synthesis and LLM-based review and scoring. The research idea, the benchmark design, the construction and audit protocol, the experimental design, the analysis code, and the interpretation of results are the authors' own work.

ETHICS STATEMENT

Benchmark tasks are reverse-synthesized from publicly available research repositories and are distributed as frozen task contracts with recorded provenance for every admitted task (Appendix D.1); we release no scraped private data. The evaluation involves no human subjects and no personally identifiable information: the only human input is the authors' own construction and win audits. Evaluated systems are accessed through their public interfaces under their terms of use, and reported per-model numbers are point estimates from a fixed protocol and budget rather than claims about a vendor's best attainable capability. More broadly, a benchmark that rewards automated method discovery could accelerate unreviewed self-improving research loops; we mitigate the immediate risk by scoring only attributable, single-component edits inside frozen evaluation harnesses, so that any measured gain remains inspectable and reproducible by a human reader.

REFERENCES

Dhruv Agarwal, Bodhisattwa Prasad Majumder, Reece Adamson, Megha Chakravorty, Satvika Reddy Gavireddy, Aditya Parashar, Harshit Surana, Bhavana Dalvi Mishra, Andrew McCallum, Ashish Sabharwal, and Peter Clark. AutoDiscovery: Open-ended scientific discovery via Bayesian surprise. In *Advances in Neural Information Processing Systems*, volume 38, pp. 28504–28542, 2025. doi: 10.52202/085713-0847. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/23b127521af7ca7a42f5cdb7507be4f2-Abstract-Conference.html.

Daniil A. Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023. doi: 10.1038/s41586-023-06792-0. URL https://doi.org/10.1038/s41586-023-06792-0.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/7e3767db483c942b883eb4f8cfb74e31-Abstract-Conference.html.

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. In *Advances in Neural Information Processing Systems*, volume 38, 2025a. doi: 10.52202/085713-3945. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/ab8dd000d6f87f40061a73f8bca7fae4-Abstract-Datasets_and_Benchmarks_Track.html. Datasets and Benchmarks Track.

Jiayao Chen, Shi Liu, and Linyi Yang. StatefulDiscovery: Evidence-calibrated claim formation in open-ended scientific discovery. *arXiv preprint arXiv:2606.11851*, 2026. URL https://arxiv.org/abs/2606.11851.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino,

Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. doi: 10.48550/arXiv.2107.03374. URL https://arxiv.org/abs/2107.03374.

Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, Vishal Dey, Mingyi Xue, Frazier N. Baker, Benjamin Burns, Daniel Adu-Ampratwum, Xuhui Huang, Xia Ning, Song Gao, Yu Su, and Huan Sun. ScienceAgentBench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *International Conference on Learning Representations*, 2025b. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/f12b4df26344f3be803c06b555252efe-Abstract-Conference.html.

Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J. R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis, and Pushmeet Kohli. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022. doi: 10.1038/s41586-022-05172-4. URL https://www.nature.com/articles/s41586-022-05172-4.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. URL https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating language agents on machine learning experimentation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 20271–20309. PMLR, 2024. URL https://proceedings.mlr.press/v235/huang24y.html.

Yuki Imajuku, Kohki Horie, Yoichi Iwata, Kensho Aoki, Naohiro Takahashi, and Takuya Akiba. ALE-Bench: A benchmark for long-horizon objective-driven algorithm engineering. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-5226. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/e555eab2c68f886052f404397ea52001-Abstract-Datasets_and_Benchmarks_Track.html. Datasets and Benchmarks Track.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida I. Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/94074dd5a072d28ff75a76dabed43767-Abstract-Conference.html.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/edac78c3e300629acfe6cbe9ca88fb84-Abstract-Conference.html.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-Tau Yih, Daniel Fried, Sida Wang, and Tao Yu. DS-1000: A natural and reliable benchmark for data science code generation. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 18319–18345. PMLR, 2023. URL https://proceedings.mlr.press/v202/lai23b.html.

Hongye Liu and Ricardo Henao. Learning to substitute words with model-based score ranking. 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. 11551–11565, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.naacl-long.576. URL https://aclanthology.org/2025.naacl-long.576/.

Hongye Liu, Liang Ding, and Ricardo Henao. Learning to control summaries with score ranking. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 35336–35359, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.1762. URL https://aclanthology.org/2026.findings-acl.1762/.

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. URL https://arxiv.org/abs/2408.06292.

Alisia Lupidi, Bhavul Gauri, Thomas Simon Foster, Bassel Al Omari, Despoina Magka, Alberto Pepe, Alexis Audran-Reiss, Muna Aghamelu, Nicolas Baldwin, Lucia Cipolina-Kun, Jean-Christophe Gagnon-Audet, Chee Hau Leow, Sandra Lefdal, Hossam Mossalam, Abhinav Moudgil, Saba Nazir, Emanuel Tewolde, Isabel Urrego, Jordi Armengol Estape, Amar Budhiraja, Gaurav Chaurasia, Abhishek Charnalia, Derek Dunfield, Karen Hambardzumyan, Daniel Izcovich, Martin Josifoski, Ishita Mediratta, Kelvin Niu, Parth Pathak, Michael Shvartsman, Edan Toledo, Anton Protopopov, Roberta Raileanu, Alexander Miller, Tatiana Shavrina, Jakob Foerster, and Yoram Bachrach. AIRS-Bench: a suite of tasks for frontier AI research science agents. *arXiv preprint arXiv:2602.06855*, 2026. URL https://arxiv.org/abs/2602.06855.

Bohan Lyu, Yucheng Yang, Siqiao Huang, Jiaru Zhang, Qixin Xu, Xinghan Li, Xinyang Han, Yicheng Zhang, Huaqing Zhang, Runhan Huang, Kaicheng Yang, Zitao Chen, Wentao Guo, Junlin Yang, Xinyue Ai, Wenhao Chai, Yadi Cao, Ziran Yang, Kun Wang, Dapeng Jiang, Huan-ang Gao, Shange Tang, Chengshuai Shi, Simon S. Du, Max Simchowitz, Jiantao Jiao, Dawn Song, and Chi Jin. MLS-Bench: A holistic and rigorous assessment of AI systems on building better AI. *arXiv preprint arXiv:2605.08678*, 2026. URL https://arxiv.org/abs/2605.08678.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6 (5):525–535, 2024. doi: 10.1038/s42256-024-00832-8. URL https://doi.org/10.1038/s42256-024-00832-8.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/70c26937fbf3d4600b69a129031b66ec-Abstract-Conference.html.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594, 2023. doi: 10.52202/075280-2019. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025. URL https://arxiv.org/abs/2506.13131.

Anne Ouyang, Simon Guo, Simran Arora, Alex L. Zhang, William Hu, Christopher Ré, and Azalia Mirhoseini. KernelBench: Can LLMs write efficient GPU kernels? In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning*

Research, pp. 47356–47415. PMLR, 2025. URL https://proceedings.mlr.press/v267/ouyang25a.html.

Rushi Qiang, Yuchen Zhuang, Anikait Singh, Percy Liang, Chao Zhang, Sherry Yang, and Bo Dai. MLE-Smith: Scaling MLE tasks with automated multi-agent pipeline. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/0abcc0198bef043543d8837cfc1e18f7-Abstract-Conference.html.

Ben Rank, Hardik Bhatnagar, Ameya Prabhu, Shira Eisenberg, Karina Nguyen, Matthias Bethge, and Maksym Andriushchenko. PostTrainBench: Can LLM agents automate LLM post-training? In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=UnjxMTe57e.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. doi: 10.1038/s41586-023-06924-6. URL https://www.nature.com/articles/s41586-023-06924-6.

Dingfeng Shi, Jingyi Cao, Qianben Chen, Weichen Sun, Weizhen Li, Hongxuan Lu, Fangchen Dong, Tianrui Qin, King Zhu, Minghao Liu, Yuchen Eleanor Jiang, Jian Yang, Ge Zhang, Jiaheng Liu, Changwang Zhang, Jun Wang, and Wangchunshu Zhou. TaskCraft: Automated generation of agentic tasks. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/48644509339cb3076f7b0407c7588af6-Abstract-Conference.html.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

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. PaperBench: Evaluating AI's ability to replicate AI research. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 56843–56873. PMLR, 2025. URL https://proceedings.mlr.press/v267/starace25a.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://papers.nips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Zhen Wang, Fan Bai, Zhongyan Luo, Jinyan Su, Kaiser Sun, Xinle Yu, Jieyuan Liu, Kun Zhou, Claire Cardie, Mark Dredze, Zhiting Hu, and Eric P. Xing. FIRE-Bench: Evaluating AI agents on the rediscovery of scientific insights. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=l3Yc9HS4US.

Zihan Wang, Lam Nguyen, Zhengyang Zhao, Mengyue Yang, Chengwei Qin, Yujiu Yang, and Linyi Yang. CreativeBench: Benchmarking and enhancing machine creativity via self-evolving challenges. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 30904–30937, San Diego, California, United States, July 2026b. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.1546. URL https://aclanthology.org/2026.findings-acl.1546/.

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*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/4f64494ecc3442f1c9261baa036378bc-Abstract-Conference.html.

Hjalmar Wijk, Tao Roa Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Joshua M. Clymer, Jai Dhyani, Elena Ericheva, Katharyn Garcia, Brian Goodrich, Nikola Jurkovic, Megan Kinniment, Aron Lajko, Seraphina Nix, Lucas Jun Koba Sato, William Saunders, Maksym Taran, Ben West, and Elizabeth Barnes. RE-Bench: Evaluating frontier AI R&D capabilities of language model agents against human experts. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 66772–66832. PMLR, 2025. URL https://proceedings.mlr.press/v267/wijk25a.html.

Hongshen Xu, Zihan Wang, Zichen Zhu, Lei Pan, Xingyu Chen, Shuai Fan, Lu Chen, and Kai Yu. Alignment for efficient tool calling of large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 17776–17792, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.898. URL https://aclanthology.org/2025.emnlp-main.898/.

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. URL https://arxiv.org/abs/2504.08066.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, pp. 50528–50652. Curran Associates, Inc., 2024. doi: 10.52202/079017-1601. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling data for software engineering agents. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3239. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/8b86cf5ace600c48fd188efbb8dedec8-Abstract-Datasets_and_Benchmarks_Track.html.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Gödel Machine: Open-ended evolution of self-improving agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/aa5f5e6eb6f613ec412f1d948dfa21a5-Abstract-Conference.html.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, and Jun Wang. Memento: Fine-tuning LLM agents without fine-tuning LLMs. *arXiv preprint arXiv:2508.16153*, 2025. doi: 10.48550/arXiv.2508.16153. URL https://arxiv.org/abs/2508.16153.

A Experimental Setup and Evaluation

Appendices A–E cover the evaluation protocol, supplementary results, trajectory and submission audits, construction evidence, and benchmark scope.

A.1 Task Coverage

Common task controls and compute. The 50 admitted tasks contain 231 public baselines and 174 settings (102 visible and 72 hidden); 29 tasks require a GPU and 21 are CPU-only. The shared task interface and fixed controls are described in §2.2.

Table 3 and Figure 6 summarize task coverage. The released `data/task_domains.csv` records all 50 task names, original categories, broad domains, innovation targets, difficulty, compute, baseline counts, and visible/hidden setting counts.

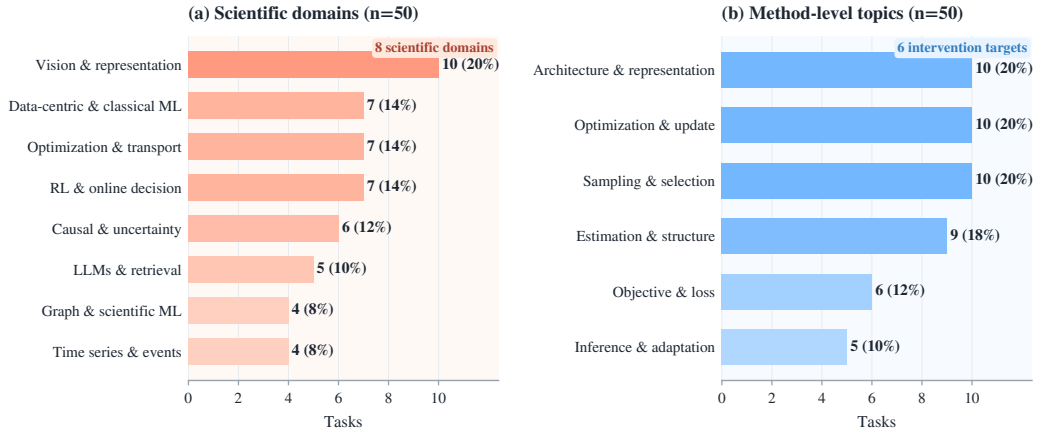


Figure 6: **The 50 tasks cover heterogeneous scientific problems and intervention types.** The left panel groups tasks into eight broad domains; the right groups them by the algorithmic component an agent may change. In-panel summaries emphasize the eight-domain and six-target coverage. Counts and percentages use the audited task-level mapping in Table 3.

Table 3: **Broad-domain inventory of the 50 evaluated tasks.** The grouping consolidates the fine-grained category stored in each task's `task.yaml`; the complete task-to-domain mapping is released with the evaluation snapshot. "GPU" counts tasks declaring nonzero GPU hours; the remaining 21 tasks are CPU-only.

	Evaluated assets					Representative scope
Broad domain	**Tasks**	**GPU**	**Baselines**	**Visible**	**Hidden**	**Task topics**
Vision & representation	10	10	47	26	15	3D Gaussian splatting, diffusion, distillation, few-shot learning, robustness
Data-centric & classical ML	7	1	36	15	11	active learning, data valuation, feature selection, anomaly detection
Optimization & transport	7	2	29	11	9	Bayesian/evolutionary/bilevel optimization, optimal transport
RL & online decision	7	4	33	15	11	offline and multi-agent RL, bandits, continual and online learning
Causal & uncertainty	6	0	26	10	7	causal discovery, conformal prediction, survival analysis
LLMs & retrieval	5	5	23	13	6	preference losses, MoE routing, KV compression, dense retrieval
Graph & scientific ML	4	3	18	4	6	message passing, community detection, materials and neural operators
Time series & events	4	4	19	8	7	forecasting, temporal attention, temporal point processes
Total	**50**	**29**	**231**	**102**	**72**	**Eight audited domains**

A.2 Discovery Protocol and Scoring

Submission and termination in Discovery. Before termination, an agent may explicitly select a previously tested candidate. A normally completed test at the test limit automatically selects that test and ends the run before another agent turn. Exhausting the action budget also retains the latest tested candidate, if any. Static-check rejections consume test budget but return control through a different path, allowing explicit selection when actions remain. Final scores therefore reflect candidate

generation, execution validity, and retention under this protocol. The `data-valuation` case in Appendix C illustrates the resulting difference in selection opportunities. All aggregate results use the candidates selected by the frozen protocol. A uniform final selection step or visible-only candidate selection would test this sensitivity; neither is evaluated here, and hidden scores are not used to reselect submissions.

Score specification and aggregation. For task t and setting j, the frozen score specification direction-normalizes objective terms to $[0, 1]$, using the weakest observed baseline as a floor and the strongest as a reference targeting 0.5, subject to bounded-score edge cases. Task-specific theoretical bounds define the scale. Weighted objective terms and constraint penalties produce a setting score $s_{t,j} \in [0, 1]$. For a valid record with at least one positive setting score, the Discovery score aggregates all m_t required visible and hidden settings as

$$D_t = \exp\left\{\frac{1}{m_t}\sum_j \log\max(s_{t,j}, 10^{-2})\right\}.$$

Valid all-zero records and records that omit or fail a required setting or validity check receive $D_t = 0$. Visible and hidden scores use the same rule on their respective setting subsets. Across tasks, Mean Discovery Score is the arithmetic mean, including invalid zeros. The common success threshold is $D_t > 0.5$; the direct baseline diagnostic uses $D_t > B_t$, where $B_t = \max_b D_t(b)$ over the included baselines. Different baselines can be best in different settings, yielding $B_t < 0.5$ after aggregation. Bounded-score edge cases can instead yield $B_t > 0.5$: in `POT`, the included `exact_emd` solver attains the theoretical optimum and scores $B_t = 1$. Thus, threshold success does not necessarily imply a strict baseline win.

A.3 Sampling and Resource Accounting

Table 4: **Observed inference and experimental-resource use.** All systems share the same 20-action and three-test ceilings, but choose different amounts of the available budget. Token totals are disclosed for transparency rather than treated as a provider-independent compute unit: tokenizers, caching, and internal request patterns differ. Foresight totals include an explanation-judge call. Actions and tests are means per Discovery cell; token columns report the cell median and full-sweep total. Foresight aggregates require complete 50-task coverage; explanation judges are not normalized across providers, so token totals remain descriptive.

	Discovery				Foresight		
Model	**Actions**	**Tests**	**Median tok.**	**Total tok.**	**Coverage**	**Median tok.**	**Total tok.**
Proprietary models							
GPT-5.4	4.18	1.62	24.6k	1.92M	50/50	6.1k	0.33M
Claude Opus 4.8	6.90	2.48	3.8k	0.22M	50/50	2.9k	0.14M
Gemini 3.1 Pro	7.26	2.90	143.6k	8.27M	50/50	15.4k	0.76M
Open-weight models							
GLM-5.2	8.00	2.86	137.1k	22.56M	50/50	16.1k	0.91M
DeepSeek-V3.2	7.08	2.78	35.1k	3.76M	50/50	7.0k	0.38M
Qwen3.5-Plus	7.56	2.82	60.3k	4.84M	50/50	7.7k	0.42M
LongCat-2.0	9.20	2.86	98.5k	7.36M	50/50	16.7k	0.82M

Uncertainty and auditable artifacts. Cross-task intervals use 20,000 task-bootstrap resamples. Paired analyses retain all model–condition observations within each sampled task. Each model–task–condition cell contributes one agent run. These intervals describe cross-task variation conditional on the recorded runs; evaluator seeds and repeated measurements of a fixed candidate are distinct from independent agent runs. For example, `data-valuation` averages three fixed seeds per setting; the audited Qwen candidates each retain 15 setting–seed logs (Appendix C). The two RQ2 comparisons pair newly collected condition cells with their frozen canonical Iterative controls. The frozen records described in §2.2 support regeneration of tables, figures, and row-level CSVs. The RQ-specific sections below document cohort membership, eligibility rules, and sensitivity analyses.

Table 4 reports observed action, test, and provider-token use. Provider token counts support auditability and vary with provider tokenizers, caching, and internal request patterns. GLM's 22.56M Discovery total is driven by a small number of very long request traces even though its median cell uses 137.1k tokens. Actions and test-harness calls provide the shared resource control; token totals are provider-specific disclosures.

B Supplementary Experimental Results

The following sections follow RQ1–RQ3 in the main text. Each groups the comparison's records, detailed results, and sensitivity checks; the final subsection compares domains and evaluation tracks.

B.1 RQ1: Discovery Outcomes and Robustness

Threshold interpretation. The frozen benchmark criterion $D_t > 0.5$ provides one common normalized threshold across tasks. The strict task-level diagnostic compares each submission with B_t, the largest aggregate score among that task's included baselines. Table 5 reports 51 threshold successes and 52 strict aggregate-baseline wins. The criteria agree on 36 model–task cells; setting aggregation and boundary-reference cases produce 15 threshold-only and 16 strict-only cells. Thus, 15 of the 51 threshold successes do not strictly exceed the strongest aggregate baseline, and the similar overall rates represent different sets of submissions. The primary analysis retains the frozen threshold; claims about exceeding an included baseline refer specifically to the strict criterion. Both criteria concern measured performance; scientific novelty requires separate assessment. Excluding the three confirmed staging-failure cells gives 51/347 threshold successes (14.7%) and 52/347 strict wins (15.0%).

Table 5: **Fixed-threshold and strict-baseline Discovery criteria.** The strict criterion compares a submission with the strongest aggregate baseline score B_t on the same task. "Both" is the intersection of the two criteria; the final columns expose their differing decisions.

	Count (rate)		Overlap (count)		
Model	Threshold $D_t > 0.5$	Strict win $D_t > B_t$	Both	Threshold only	Strict only
Claude Opus 4.8	12 (24%)	14 (28%)	11	1	3
Gemini 3.1 Pro	11 (22%)	11 (22%)	9	2	2
GLM-5.2	9 (18%)	10 (20%)	7	2	3
GPT-5.4	6 (12%)	8 (16%)	4	2	4
LongCat-2.0	6 (12%)	3 (6%)	3	3	0
DeepSeek-V3.2	4 (8%)	3 (6%)	1	3	2
Qwen3.5-Plus	3 (6%)	3 (6%)	1	2	2
All systems	**51 (14.6%)**	**52 (14.9%)**	**36**	**15**	**16**

Strict wins and measurement uncertainty. The strict criterion uses a positive observed margin $D_t - B_t$ with no minimum margin or per-submission significance requirement. Construction-time multi-seed checks establish discrimination among the included baselines; they do not establish that every submitted method's positive margin exceeds evaluation noise. A small strict win is therefore an observed comparison under the frozen evaluation, rather than a per-submission significance claim. Fresh matched-seed reevaluation of a candidate and its baseline would test margin stability beyond the fixed-seed measurements. Separately, repeated agent runs would quantify variation in which candidates each protocol produces.

Outcome-distribution decomposition. Figure 7 summarizes the complete 50-task outcome composition for each model.

Continuous Discovery score. Claude Opus 4.8 has the highest observed Mean Discovery Score (0.343). Its paired mean-score difference from Gemini 3.1 Pro is 0.047, with a 95% task-bootstrap CI of $[-0.037, 0.125]$. The interval leaves the continuous-score leader unresolved.

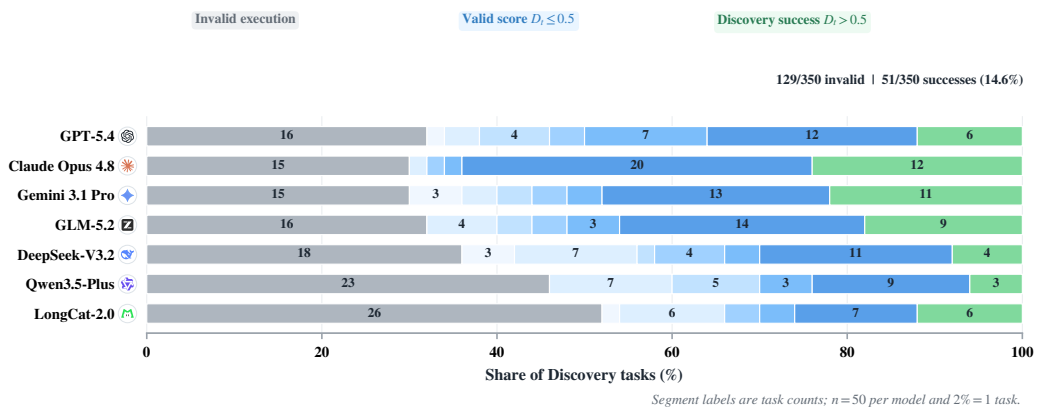


Figure 7: **Compact decomposition of Discovery outcomes.** Each bar partitions a model's 50 tasks into invalid execution (gray), valid scores up to the threshold (blue, light to dark: zero followed by 0.1-wide intervals), and Discovery successes ($D_t > 0.5$; green). Segment labels are task counts; the top annotation gives the complete-cohort totals.

Task and score sensitivity. The seven-model Discovery matrix identifies three tasks with no valid model record, one additional all-zero task, and one task on which every model scores at least 0.95. These post-hoc labels diagnose the evaluated matrix; construction-time admission remains defined by the artifact and baseline-rank gates. The released `data/task_diagnostics.csv` preserves all task-level means, dispersion, validity, and labels. Table 6 shows that removing the three all-invalid tasks, the additional all-zero task, or the empirical ceiling task preserves the model ordering. The across-task diagnostic GMean applies numerical floors to the already-computed task scores D_t; its seven-model ordering is identical for floors $10^{-3}, 10^{-6}, 10^{-9}, 10^{-12}$, and 10^{-15}, although the absolute GMean scale remains floor-dependent.

Table 6: **Discovery sensitivity to empirical floor and ceiling tasks.** Claude Opus 4.8 leads all three metrics in every row; its values are shown. The across-task diagnostic GMean applies a 10^{-9} numerical floor to the already-computed task scores D_t; removing the one ceiling task subtracts exactly one threshold success from every model.

Task set	n	Mean	GMean ($\times 10^{-4}$)	Threshold successes
All tasks	50	0.343	11.61	12/50
Exclude four floor tasks	46	0.373	39.08	12/46
Exclude one ceiling task	49	0.330	10.11	11/49
Exclude both groups	45	0.359	34.55	11/45

B.2 RQ2: Reference Context

Runtime intervention. The Reference condition supplies curated background, prior-method descriptions, baseline and provenance sections, design-space and mechanistic-axis guidance, and the visible baseline table. The no-reference condition removes this bundle through a single runtime switch. Both retain the same task objective, operational interface, source code, editable region, fixed pipeline, settings, metrics, constraints, seeds, evaluator, hidden barrier, and 20-action/three-test budget. The comparison adds this package on top of models' potential prior exposure to public papers and repositories. Campaign differences are documented below.

Coverage and provenance. All 350 Reference/no-reference pairs enter the analysis, including invalid zeros; the infrastructure sensitivity and trajectory cohort retain 347 pairs. Subset scores use the released scorer. The `reference_full` ledgers preserve model IDs, result hashes, configurations, and paired endpoints and pass protocol, pairing, and canonical-snapshot checks. Reference timestamps span August 12–31 and no-reference timestamps August 31–September 5, 2026; the raw

timestamps omit timezone offsets. Separate campaigns leave provider drift as a source of residual variation despite matched frozen controls.

Paired estimates and uncertainty. Differences are Reference minus no-reference, with equal cell weights and the task-bootstrap procedure in Appendix A. The pooled Discovery difference is $+0.0217$ (95% CI $[-0.0220, 0.0726]$), the hidden-score difference is $+0.0251$ ($[-0.0171, 0.0760]$), and the Success Rate change is $+2.57$ pp ($[-2.29, 8.29]$). The pooled intervals span zero and express substantial cross-task uncertainty. Claude's Discovery difference is $+0.0891$ ($[0.0317, 0.1476]$), and its hidden-score difference is $+0.0874$ ($[0.0200, 0.1524]$). Model-level and secondary intervals are exploratory and use nominal 95% coverage. These frozen full-matrix estimates are interpreted alongside the task-level execution checks and sensitivity analyses below.

Table 7: **Paired effect of the curated reference-context package.** Arrows show No reference to Reference on the same pairs. Reference minus no-reference defines ΔD_t; brackets give 95% task-cluster bootstrap intervals.

		Discovery score		Outcome rates	
Model	**Pairs**	**Mean score**	**ΔD_t [95% CI]**	**Success (%)**	**Execution (%)**
Pooled	350	0.230 → 0.251	+0.022 [-0.022, 0.073]	12.0 → 14.6	65.7 → 63.1
GPT-5.4	50	0.238 → 0.278	+0.040 [-0.030, 0.114]	10.0 → 12.0	74.0 → 68.0
Claude Opus 4.8	50	0.254 → 0.343	+0.089 [0.032, 0.148]	22.0 → 24.0	64.0 → 70.0
Gemini 3.1 Pro	50	0.286 → 0.296	+0.010 [-0.080, 0.104]	18.0 → 22.0	76.0 → 70.0
GLM-5.2	50	0.272 → 0.290	+0.019 [-0.059, 0.098]	12.0 → 18.0	66.0 → 68.0
DeepSeek-V3.2	50	0.186 → 0.203	+0.017 [-0.048, 0.088]	8.0 → 8.0	64.0 → 64.0
Qwen3.5-Plus	50	0.181 → 0.171	-0.011 [-0.088, 0.074]	6.0 → 6.0	60.0 → 54.0
LongCat-2.0	50	0.190 → 0.177	-0.012 [-0.098, 0.079]	8.0 → 12.0	56.0 → 48.0

The analysed cohort contains 350 matched pairs. Success requires a scoreable record with $D_t > 0.5$; execution denotes a scoreable record. Both rates use all analysed pairs. Protocol-invalid final scores equal zero. Both conditions allow 20 tool actions and three tests; the intervention toggles the complete curated reference-context package. Pooled estimates weight pairs equally; bootstrap resamples tasks with all their model pairs.

Infrastructure sensitivity. The 347-pair infrastructure sensitivity gives $\Delta D_t = +0.0219$ (95% CI $[-0.0214, 0.0735]$), preserving the direction and approximate magnitude of the complete-matrix point estimate. Table 8 gives the full paired endpoints.

POT dependency sensitivity (post hoc). Log inspection finds a campaign-specific native-extension failure in `POT`: all seven no-reference runs and the newer Claude Reference run encounter a missing `ot.lp.emd_wrap` before entering the editable solver, while six earlier Reference runs execute successfully. This task contributes $+0.0171$ (78.6%) of the pooled $+0.0217$ difference, confounding the context comparison with an execution-environment difference. We retain the frozen 350-pair primary results and report a sensitivity excluding all seven `POT` pairs. Over the remaining 343 pairs from 49 tasks, the mean difference is $+0.0048$ (95% CI $[-0.0286, 0.0399]$). This post-hoc diagnostic otherwise retains the full-matrix eligibility rules. The same task-bootstrap procedure uses analysis seed 20260911. Log inspection and this sensitivity analysis show that a campaign-specific execution failure materially affects the observed comparison. The reference package's isolated effect remains unresolved because task bootstrapping does not remove this execution-environment mismatch.

Observed search behavior. The audited 347-pair trajectory cohort yields 2.409 tested states per no-reference run and 2.470 per Reference run: a difference of $+0.061$ ($[-0.043, 0.163]$). Explicit method-composition language rises from 36.0% to 44.7%, a difference of $+8.6$ pp ($[2.0, 15.4]$). Retesting changes by -1.7 pp ($[-6.4, 3.2]$). These measurements describe code-state search and lexical tendencies, rather than establishing mechanism novelty. Qualitative human review of Discovery



Figure 8: **Paired reference-context effects and cross-task uncertainty.** Points show Reference minus no-reference changes in (a) Discovery and (b) hidden score; bars are 95% task-cluster bootstrap intervals from 20,000 resamples. Both panels use all 350 matched pairs. Diamonds mark pooled estimates; top annotations give condition means on the same cohort.

Table 8: **Transfer, execution, and interaction effects.** Changes are Reference minus no-reference. Score differences use normalized-score units; rate changes use percentage points (pp). Brackets give 95% task-cluster bootstrap intervals from 20,000 resamples.

Model	ΔHidden	ΔVisible	ΔGap	ΔExecution (pp)	ΔSuccess (pp)	ΔActions / Tests
Pooled	+0.025 [-0.017, 0.076]	+0.015 [-0.031, 0.068]	+0.010 [-0.009, 0.030]	-2.6 [-11.7, 6.6]	+2.6 [-2.3, 8.3]	+0.51 / +0.02
GPT-5.4	+0.034 [-0.047, 0.119]	+0.038 [-0.037, 0.116]	-0.004 [-0.068, 0.062]	-6.0 [-20.0, 6.0]	+2.0 [-8.0, 12.0]	+0.10 / +0.10
Claude Opus 4.8	+0.087 [0.020, 0.152]	+0.066 [0.002, 0.132]	+0.021 [-0.038, 0.086]	+6.0 [-6.0, 18.0]	+2.0 [-8.0, 12.0]	+0.32 / +0.20
Gemini 3.1 Pro	+0.028 [-0.061, 0.121]	-0.009 [-0.108, 0.093]	+0.037 [-0.015, 0.091]	-6.0 [-22.0, 8.0]	+4.0 [-10.0, 18.0]	+0.20 / -0.02
GLM-5.2	+0.007 [-0.073, 0.091]	+0.027 [-0.057, 0.115]	-0.020 [-0.083, 0.043]	+2.0 [-14.0, 18.0]	+6.0 [-4.0, 16.0]	-0.38 / -0.06
DeepSeek-V3.2	+0.013 [-0.063, 0.094]	+0.026 [-0.044, 0.101]	-0.013 [-0.070, 0.046]	+0.0 [-16.0, 16.0]	+0.0 [-8.0, 8.0]	+1.72 / -0.20
Qwen3.5-Plus	-0.012 [-0.100, 0.083]	-0.021 [-0.107, 0.072]	+0.009 [-0.064, 0.083]	-6.0 [-24.0, 12.0]	+0.0 [-10.0, 10.0]	+0.26 / -0.08
LongCat-2.0	+0.018 [-0.062, 0.104]	-0.020 [-0.111, 0.077]	+0.038 [-0.008, 0.085]	-8.0 [-24.0, 8.0]	+4.0 [-6.0, 16.0]	+1.34 / +0.20

The 347-pair infrastructure sensitivity gives $\Delta D_t = +0.022$ (95% CI [-0.021, 0.073]) and ΔHidden $= +0.025$ ([-0.017, 0.076]). Gap is the hidden-minus-visible score. Transfer endpoints use released subset aggregates across all matched cells. Primary paired score differences are positive in 139 cells, tied in 84, and negative in 127.

Table 9: **Reference context and observed search behavior.** Changes are Reference minus no-reference on the audited trajectory pairs. Tested-state changes use count units; retest, fixed-state, composition, and API-overlap changes use percentage points. Intervals are 95% task-cluster bootstrap intervals.

	Tested-code search			Static screens	
Model	**ΔStates [95% CI]**	**ΔRetest (pp) [95% CI]**	**ΔFixed (pp) [95% CI]**	**ΔCompose (pp) [95% CI]**	**ΔAPI (pp) paired coverage**
Pooled	+0.061 [-0.043, 0.163]	-1.7 [-6.4, 3.2]	-2.0 [-4.6, 0.6]	+8.6 [2.0, 15.4]	– 245/347
GPT-5.4	+0.080 [-0.220, 0.360]	+2.0 [0.0, 6.0]	+0.0 [0.0, 0.0]	+4.0 [-10.0, 18.0]	– 37/50
Claude Opus 4.8	+0.204 [-0.041, 0.449]	+0.0 [0.0, 0.0]	+0.0 [0.0, 0.0]	+8.2 [-8.2, 24.5]	– 35/49
Gemini 3.1 Pro	-0.020 [-0.143, 0.102]	+2.0 [0.0, 6.1]	+0.0 [0.0, 0.0]	+12.2 [0.0, 24.5]	– 36/49
GLM-5.2	0.000 [-0.220, 0.220]	-4.0 [-16.0, 6.0]	-2.0 [-6.0, 0.0]	+14.0 [-2.0, 30.0]	– 33/50
DeepSeek-V3.2	+0.020 [-0.280, 0.320]	-14.0 [-32.0, 4.0]	-8.0 [-24.0, 8.0]	-6.0 [-22.0, 10.0]	– 37/50
Qwen3.5-Plus	-0.122 [-0.306, 0.041]	+4.1 [-6.1, 14.3]	+0.0 [0.0, 0.0]	+18.4 [2.0, 34.7]	– 34/49
LongCat-2.0	+0.260 [-0.020, 0.540]	-2.0 [-14.0, 10.0]	-4.0 [-12.0, 4.0]	+10.0 [-8.0, 28.0]	– 33/50

The analysed cohort contains 347 pairs. Automatic finalization selects the last executed test; explicit submission selects its requested test. API coverage counts pairs with measurable symbol-aligned screens in both conditions. A dash denotes an unavailable all-pair estimate. State and lexical features describe observable behavior; scientific mechanisms are audited separately.

findings is reported in Appendix C.3. The baseline API screen is measurable in both conditions for 245/347 pairs. The table reports that coverage and marks the all-pair API estimate as unavailable. Each analysed submission is linked to its scored test using accepted edits, undo operations, explicit selection, and automatic finalization.

B.3 RQ2: One-shot and Iterative Protocols

Interaction ceilings and comparison scope. One-shot permits two tool actions and one official test; Iterative permits 20 actions and three tests. The action ceiling is ten times larger, while the test ceiling is three times larger. RQ2's experimentation comparison uses two bounded protocols that jointly change feedback, revision opportunities, and interaction budget. The submission rules in Appendix A determine which candidate is scored.

Paired estimates and uncertainty. The comparison uses all 350 matched model–task pairs. Differences are Iterative minus One-shot, with equal pair weights and 95% percentile intervals from the shared task-bootstrap procedure. The pooled Mean Discovery Score difference is $+0.0205$ (95% CI $[-0.0189, 0.0585]$).

Figure 3 visualizes the comparison; its score and success-rate axes start at 0.20 and 10%, with marked breaks, while the cost axis starts at zero. Annotations use unrounded means.

Paired success outcomes. Among all 350 pairs, 22 succeed under both protocols, 29 only under Iterative, 26 only under One-shot, and 273 under neither. Of the 55 pairs with different success outcomes, 37 have scoreable submissions in both conditions (19 Iterative-only successes and 18 One-shot-only successes); the other 18 contain an unscoreable outcome (10 and eight, respectively). These single-run pairs describe outcome variation under the two protocols, including sampling effects.

Full model-level comparison. Table 10 reports condition means, score differences, Success Rates, and Execution Rates for every model. The pooled comparison contains 144 positive, 82 tied, and 124 negative score differences. Four model means increase and three decrease; all seven model-level score intervals include zero. The pooled hidden-score difference is $+0.0154$ ($[-0.0247, 0.0544]$). Mean recorded actions are 1.99 versus 7.17, and test calls are 0.92 versus 2.62. Action counts include dispatched edit, undo, test, and explicit submit calls; test counts include calls that reach static checks.

Table 10: **Full-matrix One-shot–Iterative comparison.** Arrows show One-shot to Iterative on the same pairs. Iterative minus One-shot defines ΔD_t; brackets give 95% task-cluster bootstrap intervals.

		Discovery score		Outcome rates	
Model	**Pairs**	**Mean score**	**ΔD_t [95% CI]**	**Success (%)**	**Execution (%)**
Pooled	350	0.231 → 0.251	+0.021 [-0.019, 0.059]	13.7 → 14.6	65.4 → 63.1
GPT-5.4	50	0.221 → 0.278	+0.057 [-0.008, 0.122]	12.0 → 12.0	74.0 → 68.0
Claude Opus 4.8	50	0.287 → 0.343	+0.056 [-0.047, 0.157]	18.0 → 24.0	74.0 → 70.0
Gemini 3.1 Pro	50	0.321 → 0.296	-0.025 [-0.097, 0.051]	24.0 → 22.0	82.0 → 70.0
GLM-5.2†	50	0.210 → 0.290	+0.080 [-0.003, 0.165]	10.0 → 18.0	66.0 → 68.0
DeepSeek-V3.2	50	0.162 → 0.203	+0.042 [-0.020, 0.105]	6.0 → 8.0	50.0 → 64.0
Qwen3.5-Plus	50	0.224 → 0.171	-0.054 [-0.129, 0.023]	14.0 → 6.0	60.0 → 54.0
LongCat-2.0	50	0.190 → 0.177	-0.013 [-0.065, 0.037]	12.0 → 12.0	52.0 → 48.0

The analysed cohort contains 350 matched pairs. Success requires a scoreable record with $D_t > 0.5$; execution denotes a scoreable record. Both rates use all analysed pairs. Protocol-invalid final scores equal zero. One-shot allows two tool actions and one test; Iterative allows 20 actions and three tests. Both retain the reference context. Pooled estimates weight pairs equally; bootstrap resamples tasks with all their model pairs. † GLM's One-shot runtime explicitly disables thinking; its earlier control omits that request field. Appendix B.3 reports the six-model sensitivity.

Table 11: **Transfer, execution, and interaction effects.** Changes are Iterative minus One-shot. Score differences use normalized-score units; rate changes use percentage points (pp). Brackets give 95% task-cluster bootstrap intervals from 20,000 resamples.

Model	**ΔHidden**	**ΔVisible**	**ΔGap**	**ΔExecution (pp)**	**ΔSuccess (pp)**	**ΔActions / Tests**
Pooled	+0.015 [-0.025, 0.054]	+0.023 [-0.020, 0.064]	-0.008 [-0.033, 0.018]	-2.3 [-12.9, 7.7]	+0.9 [-3.4, 5.1]	+5.17 / +1.69
GPT-5.4	+0.052 [-0.022, 0.123]	+0.064 [-0.010, 0.137]	-0.012 [-0.062, 0.036]	-6.0 [-20.0, 8.0]	+0.0 [-8.0, 8.0]	+2.18 / +0.62
Claude Opus 4.8	+0.059 [-0.041, 0.158]	+0.037 [-0.074, 0.145]	+0.022 [-0.027, 0.074]	-4.0 [-22.0, 14.0]	+6.0 [-8.0, 20.0]	+4.90 / +1.54
Gemini 3.1 Pro	-0.009 [-0.073, 0.059]	-0.042 [-0.130, 0.048]	+0.032 [-0.025, 0.093]	-12.0 [-26.0, 0.0]	-2.0 [-18.0, 14.0]	+5.28 / +1.98
GLM-5.2†	+0.045 [-0.042, 0.133]	+0.104 [0.018, 0.194]	-0.059 [-0.116, -0.006]	+2.0 [-16.0, 20.0]	+8.0 [-6.0, 22.0]	+6.00 / +2.02
DeepSeek-V3.2	+0.031 [-0.036, 0.098]	+0.060 [-0.013, 0.134]	-0.029 [-0.087, 0.026]	+14.0 [-4.0, 32.0]	+2.0 [-4.0, 10.0]	+5.10 / +1.86
Qwen3.5-Plus	-0.070 [-0.147, 0.008]	-0.045 [-0.128, 0.038]	-0.025 [-0.084, 0.029]	-6.0 [-24.0, 10.0]	-8.0 [-18.0, 0.0]	+5.56 / +1.86
LongCat-2.0	0.000 [-0.058, 0.060]	-0.017 [-0.077, 0.041]	+0.017 [-0.034, 0.066]	-4.0 [-20.0, 12.0]	+0.0 [-6.0, 6.0]	+7.20 / +1.98

The 347-pair infrastructure sensitivity gives $\Delta D_t = +0.024$ (95% CI [-0.015, 0.062]) and $\Delta\text{Hidden} = +0.018$ ([-0.021, 0.057]). Gap is the hidden-minus-visible score. Transfer endpoints use released subset aggregates across all matched cells. Primary paired score differences are positive in 144 cells, tied in 82, and negative in 124.

Coverage and provenance. The One-shot release (`e2d29de`, runtime `fa61c421`) contains 350 final results and 354 checksum-verified payloads, with unchanged task definitions and scoring assets. All 350 pairs enter the analysis, including invalid zeros and the released visible/hidden subset scores. Cell ledgers preserve result hashes, model IDs, action/test counts, visibility signatures, and exclusions. The infrastructure-filtered sensitivity gives $\Delta D_t = +0.0236$ (95% CI $[-0.0154, 0.0617]$); it remains separate from the configuration sensitivity below.

Scheduling and runtime-configuration sensitivity. The full One-shot campaign ran from August 31 to September 5, 2026 (UTC), after the canonical evaluations whose raw result timestamps span August 12–31. The imported runtime includes dependency, isolated-seed-workspace, scheduling, and timeout fixes. For GLM-5.2, it also explicitly requests disabled thinking, whereas the earlier control omits that request field; the earlier gateway default is unrecorded. Tables mark this model with a dagger. A configuration sensitivity combines the audited infrastructure filter with exclusion of all 50 GLM pairs, retaining 297 pairs across 50 tasks: $\Delta D_t = +0.0141$ ($[-0.0249, 0.0512]$), ΔHidden $= +0.0139$ ($[-0.0254, 0.0520]$), and Δ(Hidden–Visible) $= +0.0009$ ($[-0.0251, 0.0271]$). The primary and both sensitivity analyses yield positive pooled score estimates with intervals spanning zero. Single-run sampling and the runtime differences above leave the isolated effect of additional experimentation unresolved, including the value of longer trajectories with uniform candidate retention.

B.4 RQ3: FORESIGHT

Rankings, flags, and mechanisms. Foresight requires one best-first method ranking and per-method mechanism explanations, with optional method-level OOD-sensitivity flags. Overall Pairwise Ranking Accuracy evaluates the ranking against the overall order, and OOD-Sensitive Method F1 evaluates the flags against methods involved in a strict overall-to-hidden inversion. The same ranking's agreement with the hidden-only order supplies an alignment diagnostic. Best-Method Accuracy measures the first-ranked choice, Balanced Accuracy weights sensitivity and specificity equally, and flag coverage records output completeness. Missing flags count as negative predictions. Explanation quality and the composite score are reported below. Table 12 and Figure 9 report the complete deterministic decision profile, including Best-Method Accuracy, hidden-set alignment, Balanced Accuracy, and flag coverage.

Table 12: **Method ranking and OOD-sensitivity results.** Overall Pairwise Ranking Accuracy measures broad method ordering, and OOD-Sensitive Method F1 measures method-level sensitivity identification; these are the two primary outcomes. Hidden-set alignment, Best-Method Accuracy, Balanced Accuracy, and flag coverage contextualize those outcomes.

	Overall ranking		Hidden ranking (diagnostic)		OOD-sensitive methods		
Model	**Pairwise accuracy↑**	**Best-method accuracy ↑**	**Pairwise accuracy↑**	**Best-method accuracy ↑**	**F1↑**	**Balanced accuracy↑**	**Flag coverage**
GPT-5.4	<u>68.9%</u>	42.0%	65.8%	44.0%	31.8%	52.3%	51.5%
Claude Opus 4.8	66.8%	38.0%	65.4%	44.0%	<u>35.1%</u>	52.0%	100%
Gemini 3.1 Pro	67.1%	40.0%	65.6%	50.0%	**38.3%**	53.8%	100%
GLM-5.2	**69.2%**	46.0%	67.9%	54.0%	31.7%	50.5%	100%
DeepSeek-V3.2	65.8%	44.0%	64.5%	50.0%	30.2%	52.0%	100%
Qwen3.5-Plus	64.9%	38.0%	64.6%	44.0%	35.0%	55.9%	98.3%
LongCat-2.0	**69.2%**	40.0%	68.7%	46.0%	34.1%	53.5%	93.5%

Reference values. Random ranking gives 50.0% pairwise accuracy in both views and 23.9%/25.0% overall/hidden Best-Method Accuracy. Flagging every method gives 44.4% F1, 50.0% Balanced Accuracy, and 100% flag coverage.

Pairwise Accuracy excludes ground-truth ties; Best-Method Accuracy is tie-aware. An OOD-sensitive method participates in at least one strict overall-to-hidden rank inversion. Every model is evaluated against the same ground-truth set: 66/231 task–method instances across 25/50 tasks are positive. Ranking metrics are task-macro averages; F1 and Balanced Accuracy pool task–method decisions. Missing flags count as negative predictions. Bold and underlined values mark the highest and second-highest primary point estimates; displayed ties share a style.

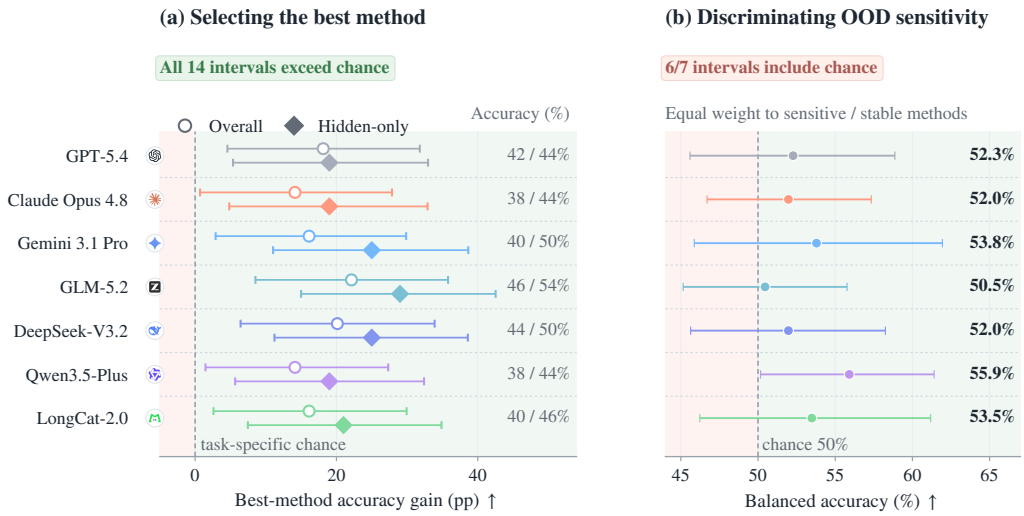


Figure 9: **Method selection exceeds chance while OOD-sensitive-method discrimination remains near chance.** (a) Best-Method Accuracy minus task-specific chance for the overall order and the hidden-set alignment diagnostic; all 14 intervals exceed zero. (b) Balanced Accuracy for the submitted OOD-sensitivity flags; the dashed line marks 0.5 and six of seven intervals include it. Error bars are 95% task-bootstrap intervals.

Table 13: **Mechanism explanations and composite Foresight scores.** Each model has 50 scored tasks. Means of the normalized correlation component R, explanation quality E, and composite $F = 0.6R + 0.4E$ are reported as descriptive diagnostics.

Model	Mechanisms provided	R	E	F	95% CI for F
GPT-5.4	231/231	0.693	0.481	0.608	[0.552, 0.662]
Claude Opus 4.8	231/231	0.677	0.364	0.552	[0.498, 0.606]
Gemini 3.1 Pro	231/231	0.675	0.413	0.570	[0.518, 0.620]
GLM-5.2	230/231	0.701	0.482	0.613	[0.561, 0.664]
DeepSeek-V3.2	231/231	0.665	0.427	0.570	[0.511, 0.628]
Qwen3.5-Plus	231/231	0.656	0.517	0.600	[0.542, 0.657]
LongCat-2.0	222/231	0.702	0.473	0.610	[0.548, 0.671]

GPT-5.4 scores all explanations and is also evaluated as a solver. Mechanism coverage counts non-empty explanations matched to the submitted method names. These are task-level judge scores, not averages of per-method scores. Intervals use 20,000 task-bootstrap resamples; they do not quantify judge bias.

Mechanism judgment and composite scoring. The prediction prompt requests an explanation for every candidate, and each explanation object requires a `mechanism` field. The judge receives the submitted explanations and the true overall and hidden-only method orders, and returns one task-level $E_t \in [0, 1]$ for mechanistic correctness, insight, and anticipation of OOD reversals. This is an LLM assessment of the submitted reasoning, not an experimental verification of the proposed mechanisms.

Let $\mathcal{V}_t$ contain the overall ranking and, when available, the hidden-only ranking. For each view v, the evaluator computes Spearman's $\rho_{t,v}$ and Kendall's $\tau_{t,v}$ for the same submitted ranking. The released score normalizes their mean before combining it with E_t:

$$R_t = \frac{1}{2}\left(1 + \frac{1}{|\mathcal{V}_t|}\sum_{v\in\mathcal{V}_t}\frac{\rho_{t,v}+\tau_{t,v}}{2}\right), \qquad F_t = 0.6R_t + 0.4E_t.$$

Thus R_t is a correlation-based component, not the deterministic Pairwise Ranking Accuracy in the primary analysis; OOD-Sensitive Method F1 is also reported separately. If the judge is unavailable, the evaluator retains a rank-only partial record and leaves F_t missing rather than treating the missing explanation score as zero.

Coverage and judge provenance. The selected cohort contains 300 records from the six original non-Claude systems and 50 from Claude Opus 4.8, excluding the superseded Opus 4.7 campaign. All 350 have finite E_t and F_t satisfying the formula above. Of 1,617 requested explanations, 1,607 contain mechanism text: GLM omits one field and LongCat supplies empty lists on two tasks covering nine methods. Their observed task-level judge scores are retained.

GPT-5.4 is the explanation judge for all seven systems and is also evaluated as a solver. Table 13 reports the explanation and composite scores as descriptive diagnostics. Task-bootstrap intervals quantify cross-task variation conditional on this judge; they do not calibrate judge bias. Overall Pairwise Ranking Accuracy and OOD-Sensitive Method F1 remain the primary cross-model outcomes.

Public-source exposure. Official repositories and baseline papers are public, making pretraining exposure plausible. Foresight retains method identities and descriptions, so ranking can draw on prior knowledge of published methods as well as the provided task information. Above-random ranking therefore establishes pre-execution predictive ability in this information setting, without isolating mechanism understanding from prior familiarity. Method-name-masked cards and temporally held-out tasks would test this distinction; neither control is included in the reported evaluation.

B.5 Across-task and Cross-track Comparisons

Domain heterogeneity. Figure 10 shows that the Discovery point-estimate leader changes by domain. GLM-5.2 leads vision and optimization; Claude Opus 4.8 leads data-centric ML, RL, causal/uncertainty, and graph/scientific ML; GPT-5.4 leads LLMs/retrieval jointly with Claude; and Gemini leads time series. The Foresight panel applies the deterministic Pairwise Ranking Accuracy used by RQ3. With 4–10 tasks per domain and no multiplicity correction, both panels provide hypothesis-generating comparisons.

Visible–hidden transfer and hidden-set alignment. Figure 11 compares each model's hidden-minus-visible Discovery gap and the hidden-only-minus-overall Pairwise Ranking Accuracy of its single Foresight ranking. All seven Discovery intervals and all seven Foresight intervals include zero. The point estimates vary in sign for Discovery and are slightly negative for Foresight, while the paired uncertainty leaves both model-level transfer orderings unresolved.

Cross-track association. Across seven models, Mean Discovery Score and deterministic Overall Pairwise Ranking Accuracy have Spearman $\rho = 0.25$ (exact two-sided permutation $p = 0.595$). This weak descriptive association leaves open whether code-based Discovery and execution-free ranking draw on a shared model capability.

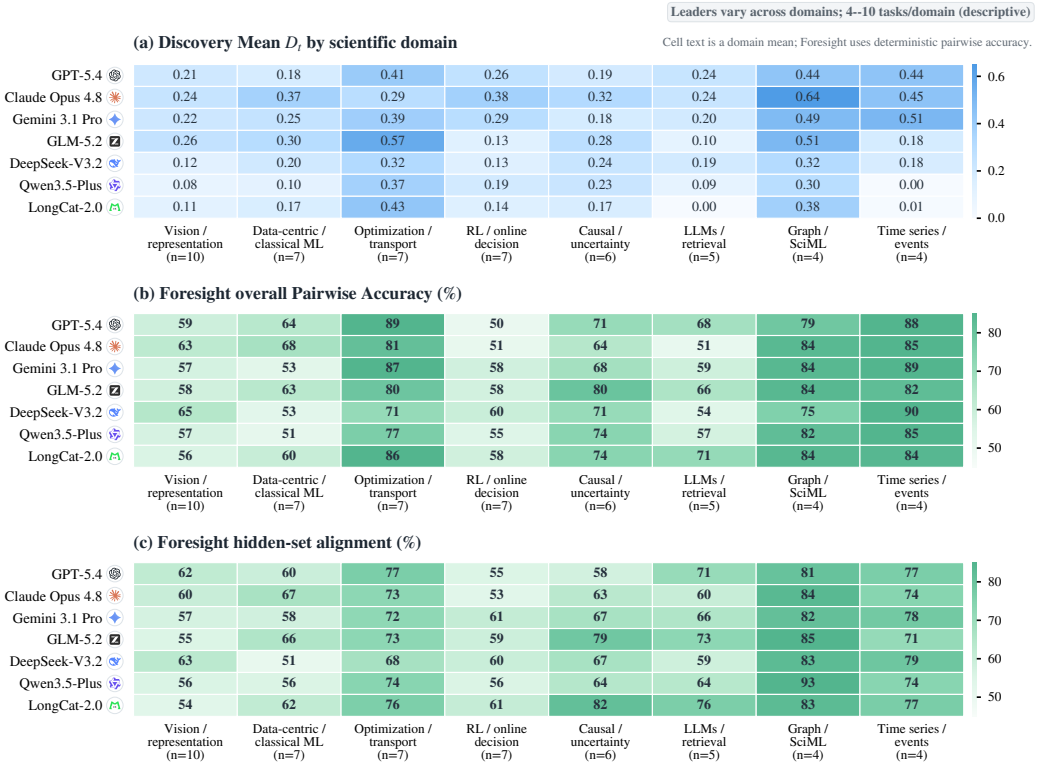


Figure 10: **Domain-level Discovery and Foresight performance is heterogeneous.** Discovery cells report Mean Discovery Score; Foresight cells report Overall Pairwise Ranking Accuracy and hidden-set alignment, averaged over tasks in each scientific domain. Column labels give domain sample sizes. All models have complete 50-task coverage; groups contain 4–10 tasks, so the panel supports descriptive domain diagnosis.

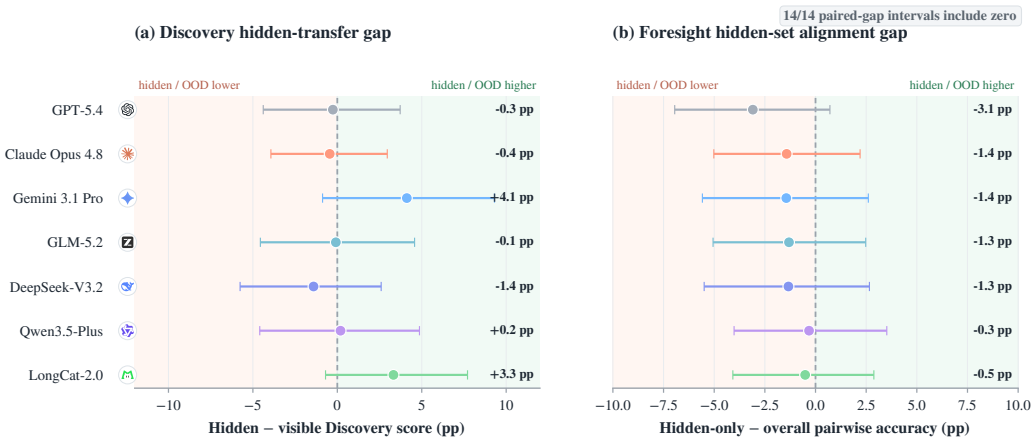


Figure 11: **Main-sweep visible/hidden and hidden-alignment diagnostics.** Points are paired hidden-minus-visible Discovery gaps and hidden-only-minus-overall Pairwise Ranking Accuracy gaps. Bars are 95% task-bootstrap CIs; all 14 intervals include zero. Shading identifies the direction of each gap.

C TRAJECTORY CASES AND SUBMISSION AUDITS

This appendix documents the 350-run replay, selected mechanism cases, and the status of the threshold-success audit.

C.1 TRAJECTORY RECONSTRUCTION AND BEHAVIOR PROFILES

Trajectory reconstruction. The full-trace audit reconstructs the complete editable surface from the initial prompt, advances it after an `OK` edit result, restores successful `undo` snapshots, and records a candidate when the tool returns an executed `TEST #`. Calls that fail static checks still count as tests and can introduce a new workspace state; only calls that reach evaluation produce performance measurements. This rule excludes 46 test attempts rejected after budget exhaustion. Candidate identity hashes the full editable surface, and `submit(n)` binds the analysis to the selected test. A run ending without an explicit selection automatically submits its last executed test; the replay implements that same rule. The replayed code state matches the compact archive in 326/350 cells. Twenty-four legacy archives retain a later workspace edit because the early runtime selected test metrics without restoring its code snapshot. The released replay recovers the score-bound code for those cells, including two Discovery successes; aggregate scores remain unchanged.

All 350 completed runs enter the case-study statistics, including their frozen outcomes and recorded test calls. Three `pbgd-hypergrad` runs record a missing editable file and a final score of zero. The workspace hash includes file absence, so their unchanged workspace states remain part of the activity counts. The audit retains the setup-failure flags for infrastructure diagnostics. It also applies a fixed lexicon for explicit composition language to code introduced by the selected candidate. The resulting `data/discovery_trajectory_audit.csv` covers all 350 cells, and `data/discovery_trajectory_summary.json` freezes the aggregate counts. The lexical marker supports reproducible case selection and remains sensitive to coding style and task mix.

Behavior profiles and contrasts. Table 14 and Figure 12 give the complete model profiles and overlapping behavior groups supporting Section 3.4. Among valid submissions, those with explicit composition language succeed in 13/99 cases (13.1%), compared with 38/122 (31.1%) without it. One-state trajectories yield 13/66 successes (19.7%); 41 trajectories retest a prior state and yield two successes, while all 17 fixed-state trajectories fail. These are descriptive associations: task/model mix, overlapping groups, and stopping rules prevent causal attribution.

Table 14: **Models exhibit distinct Discovery search profiles.** All 350 completed Iterative runs are included. Every model has $N = 50$; all percentages use this same denominator.

		Interaction path		Workspace-state search				Candidate form		Outcome	
Model	N	Direct path	All 3 tests	One state	≥3 states	Retest	Fixed	Explicit composition	Edit distance	Execution Rate	Discovery Success Rate
GPT-5.4	50	52%	20%	58%	18%	2%	0%	48%	0.266	68%	12%
Claude Opus 4.8	50	6%	58%	10%	58%	0%	0%	38%	0.113	70%	**24%**
Gemini 3.1 Pro	50	0%	90%	2%	88%	4%	2%	20%	0.218	70%	<u>22%</u>
GLM-5.2	50	6%	92%	6%	86%	6%	0%	52%	0.171	68%	18%
DeepSeek-V3.2	50	4%	88%	40%	42%	48%	30%	46%	0.113	64%	8%
Qwen3.5-Plus	50	4%	90%	8%	82%	8%	0%	48%	0.245	54%	6%
LongCat-2.0	50	0%	92%	8%	78%	14%	2%	58%	0.197	48%	12%

Direct path denotes the exact `edit`→`test`→`submit` sequence. Retest records a recurring workspace state; Fixed records the same state across multiple executed tests. State hashes include file absence. Explicit composition is a lexical screen over introduced code. Edit distance is the median normalized-token distance between consecutive tested states in multi-test runs. Bold/underline mark the first/second Success Rate.

C.2 CANDIDATE RETENTION AND MECHANISM CASES

Same-task contrast: revision validity and finalization. On `data-valuation`, Claude's initial KNN-Shapley/Banzhaf blend produces visible score 0.527, hidden score 0.489, and $D_t = 0.512$. Its next two test calls fail static checks and return no performance measurements. The static failure branch returns control to the agent, which later submits h_1. Qwen's first candidate exceeds the utility budget; its exact KNN-Shapley h_2 and distance-weighted h_3 both complete evaluation. The third-test

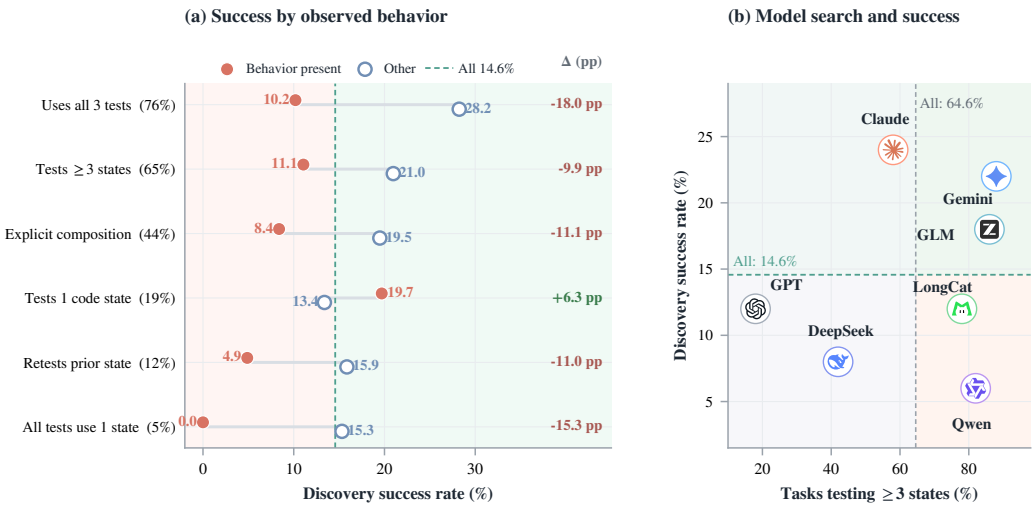


Figure 12: **Expanded behavior contrasts and model search profiles.** The six behavior groups overlap. Filled endpoints show each group's success rate and outlined endpoints show the complementary group; row labels give prevalence. The right panel reports model-level search breadth and success. All 350 runs are included. Test calls include static-check rejections but exclude calls refused after budget exhaustion. Comparisons are descriptive.

execution path selects the last test and ends the run before a further agent turn. This shared rule creates different feedback and selection opportunities along the successful-execution and static-rejection paths.

Re-scoring the 15 saved setting/seed logs for each Qwen candidate gives visible/hidden/overall scores of $0.359/0.457/0.396$ for h_2 and $0.079/0.332/0.141$ for h_3. Two available task snapshots have identical score specifications and baseline anchors; both reproduce the published h_3 scores exactly. These are retrospective scores of recorded tests, with hidden metrics available to the analyst but withheld from agent feedback. Retaining the recorded h_2 would recover 0.255 score, while leaving the submission below the 0.5 threshold. Thus, available-candidate quality and budget-end retention are separate contributors to the observed outcome. The local case audit records the result and log checksums and the rescoring procedure.

Mechanism contrasts. Table 15 summarizes six purposively selected cases. For `cdlib`/DeepSeek, three tests of the same spectral–KMeans code with identical seeds yield identical metrics. Under the hidden high-mixing stressor, communities fall from 19.33 to 1.33 and normalized performance from 0.657 to 0.105. The task brief identifies this regime as risky for the neighbor-majority polishing step; its causal role requires a matched ablation. The `POT`/GPT-5.4 case clears the fixed threshold by calling the existing `ot.emd` solver while tying the strongest aggregate baseline. These cases identify interventions, not their population frequency or novelty; mechanism ablations and fixed-candidate retention studies remain necessary.

C.3 Success-Submission Audit

Discovery-success artifact triage. The released `data/discovery_win_inventory.csv` enumerates all 51 Discovery-success submissions with their visible/hidden scores, resource use, and edited filenames. Table 16 and `data/discovery_win_artifact_audit.csv` add a conservative static screen over the corresponding submitted edit symbols. All 51 target regions are recoverable. The screen finds 0/51 token-for-token copies of a complete public baseline edit and 0/51 references to the frozen evaluation artifacts. Eleven submissions cross the sensitive 50% contiguous-coverage threshold, 20 share a task-distinctive public-baseline API identifier, and seven come from the single empirical ceiling task. These flags are screening signals for mechanism review, not determinations of reuse or novelty.

Table 15: **Six trajectory-grounded Discovery cases.** The cases contrast local mechanism search, fixed-state measurement, named-method composition, repository-native reuse, and hidden-set retention. States count distinct complete versions submitted to the test harness; test calls include static-check failures and exclude calls rejected after budget exhaustion. Visible and hidden values are normalized scores.

Pattern	Task / model	Observed tested search	States / tests	Visible → hidden	Case finding
Repository-native reuse	`POT` / GPT-5.4	Calls the exposed exact `ot.emd` primitive	1/1	1.000 → 1.000	This case clears the fixed threshold but ties the strongest aggregate baseline, showing that the two criteria select different events.
Local revision	`MAPIE` / LongCat	Tests three closely related out-of-fold calibrated CQR states	3/3	.491 → .479	The code changes remain within one calibration family and finish below the calibrated success threshold ($D_t = .485$).
Named-method composition	`SMPyBandits` / Gemini	Tests Thompson-style, KL-UCB, then a clipped combination	3/3	.520 → .511	Adjacent named mechanisms clear both the fixed threshold and the strict baseline criterion ($D_t = .516$).
Local enumeration	Bayesian optimization / GLM	Revises LHS/trust-region search with EI, UCB, and posterior-ranked variants	3/3	.565 → .136	Three code states stay in one acquisition-policy neighborhood, while normalized score drops from .565 visible to .136 hidden ($D_t = .319$).
Mechanism mismatch	`cdlib` / DeepSeek	Repeats one spectral–KMeans state with up to eight neighbor-majority sweeps	1/3	.657 → .105	Under high mixing, the final majority-polished output detects 1.33 communities versus 19.33 on the visible setting; three tests use the same seeds and return identical metrics ($D_t = .262$).
Hidden-retaining intervention	Dense retrieval / GPT-5.4	Selects negatives using query hardness, positive dissimilarity, and diversity	1/1	.746 → .801	The task-conditioned selector retains high performance on the held-out ArguAna domain ($D_t = .764$).

Corpus frequencies use all 350 completed Iterative runs; these six purposively selected cases provide mechanism contrast.

Table 16: **Static audit of the 51 threshold successes and human-review status.** Comparisons use only the submitted definition corresponding to each public baseline's declared edit symbol, avoiding repository-native code outside the agent's edit surface. Token matching ignores whitespace and comments but retains identifiers and literals. A distinctive API identifier is unique to one baseline within that task after generic language, array, and training operations are removed. Coverage and identifier overlap are screening signals, not novelty verdicts. The completed human review is summarized qualitatively in the accompanying text.

Auditable check	Observed	Interpretation
Coverage and concentration		
Threshold-success cases indexed	51 submissions / 19 tasks	Every $D_t > 0.5$ cell is enumerated
Submitted target regions recovered	51/51	Static comparisons cover the complete threshold-success set
Threshold successes on one empirical ceiling task	7/51 (13.7%)	Concentration on `matgl-crystal-net` informs interpretation
Conservative static screens		
Exact public-baseline token containment	0/51	No target contains a complete normalized baseline token sequence
Longest contiguous baseline coverage $\geq 50\%$	11/51	Short common wrappers can trigger this screen
Distinctive public-baseline API identifier overlap	20/51	API overlap alone does not establish mechanism equivalence
Evaluation-artifact reference	0/51	No target region names the frozen score/rank artifacts screened
Completed qualitative human review		
Human validity and mechanism review	Completed	Two researchers support result validity and methodological value

Completed human review and interpretation. Two researchers completed a manual review of reported Discovery findings, including high-scoring and threshold-success submissions. They checked submission–result correspondence, edit-scope compliance, and evaluation shortcuts, distinguishing known-method reuse from substantive mechanism changes. They confirmed the credibility of the inspected performance results, found no shortcuts that would invalidate them, and identified substantive mechanism differences in some submissions. This review supports the methodological value of the component-level Discovery findings. We report the review qualitatively, without a per-submission novelty rate or reviewer-agreement statistic. Threshold success remains a performance measure, not a determination of mechanism novelty. Cohort-wide matched reuse and configuration controls are not included. The static audit inventory covers the 51 threshold successes, rather than all 52 strict baseline wins.

D SYNTHESIS PIPELINE EVIDENCE

This appendix section gives the construction-audit provenance, full stage contracts, exit-gate thresholds, task format, and verdict taxonomy for the open-ended discovery pipeline of §2.3.

D.1 CONSTRUCTION-AUDIT PROVENANCE

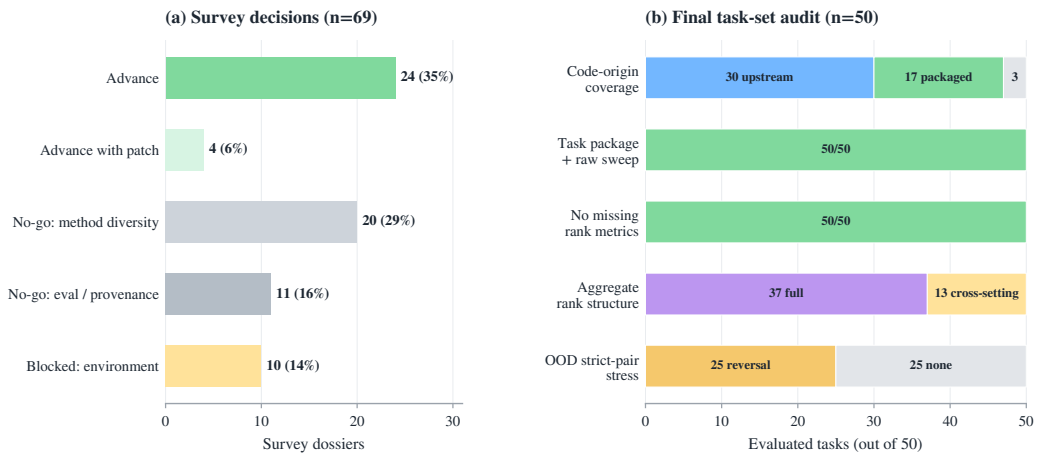


Figure 13: **Artifact-backed construction audit.** (a) Among 69 Survey dossiers, 28 advance directly or with a compatibility patch; the remaining 41 stop at method-diversity, evaluation/provenance, or environment gates. (b) In the final set, 30 tasks resolve to independent upstream Git roots, 17 are self-contained literature-grounded packages, and three external checkouts remain unresolved. All 50 retain a complete package, raw baseline sweep, and rank report; 37 fully separate aggregate ranks, and 25 contain a strict overall-to-hidden pairwise reversal.

Figure 13 uses two frozen checkpoints. The Survey collection contains 61 pinned remote repositories and eight local scaffolds with source and environment metadata; dossiers record each top-line verdict and primary stop gate. The final-set collection uses the 50 evaluated task manifests, raw baseline sweeps, and rank reports. Row-level evidence is released in `data/pipeline_survey.csv` and `data/task_quality_audit.csv`. The pipeline supplies the task definitions, research questions, and baseline lists. These overlapping snapshots lack a persistent identifier across every stage, so they support checkpoint counts rather than end-to-end yield.

The code-origin audit (`data/task_provenance.csv`) requires a code root to be its own Git top level, avoiding attribution to a parent repository. Table 17 reports 30 verified upstream checkouts and 17 self-contained, literature-grounded packages. Three provenance gaps remain: `3dgs-optimization` and `batch-active-learning` have absent declared roots, while `SMPyBandits` resolves only to a parent Git root. All three retain their task packages and evaluation records. Structured paper references occur in 227/231 baseline edit files, including 153 arXiv identifiers.

Table 17: **Code-origin audit of the final 50 tasks.** Artifact auditability is assessed separately from external-checkout provenance.

Code-root mode	Tasks	Evidence
Verified upstream Git	**30**	Root, origin, and HEAD
Self-contained package	**17**	Frozen benchmark artifact
Declared root absent	2	Declared path, no checkout
Parent Git only	1	Independent root unresolved

Within the evaluated set, all 50 task packages retain raw baseline evidence and rank reports without missing metrics. Thirty-seven reports have a fully distinct aggregate ordering. The other 13 retain aggregate rank ties created by setting-specific method-order crossings, providing direct generalization stress tests. The tie-aware overall-to-hidden audit finds at least one strict pairwise inversion on 25 tasks, comprising 40/429 method pairs and involving 66/231 methods. The overall and hidden-only best sets differ on 15 tasks and are disjoint on seven.

Construction cost and human involvement. The two checkpoints do not define an end-to-end construction yield. Historical intermediate-stage retries, model calls, token use, and human hours were retained unevenly, preventing a complete cost estimate. Human review is part of the workflow specification, but per-task intervention counts and review time are unavailable. Accordingly, automated synthesis describes artifact generation by the pipeline; its reduction in manual work and its cost of scaling remain unmeasured. The resource totals in Table 4 describe evaluated agents' task attempts and cannot supply these construction costs.

D.2 Stage Contracts and Exit Gates

Orchestration and recovery. Stages proceed in the order below. The orchestrator checks exit conditions and routes recovery; stage agents generate the artifacts. A PASS advances, a fixable FAIL retries the same stage with feedback (up to three retries), and a structural failure returns to scout without relaxing scope constraints.

Source-setting fidelity. Upstream checkouts reuse a verified source evaluation, while self-contained packages freeze a literature-grounded evaluation protocol. Official simulations are admissible: examples include SHD on simulated DAGs for causal discovery and hypervolume on ZDT/DTLZ for multi-objective optimization. Settings must follow the source experiments rather than be adjusted to manufacture method separation (Liu & Henao, 2025; Liu et al., 2026). Construction anchors on a unified multi-method library where available, reuses the source configuration, reproduces before porting, and checks multi-seed rank fidelity after the port.

D.3 Task Artifacts and Rank Verdicts

Each `task.yaml` freezes task metadata and provenance (including a code root and canary identifier), a single edit range per editable file, creation constraints and seeds, visible/hidden evaluation commands, hidden metrics, baseline edits, and the scoring parser/specification. Baseline `.edit.py` files expose a module-level `OPS` list restricted to `replace` and `create` operations.

Post-sweep rank checks assign *qualified* when aggregate ranks fully separate; *kept* when aggregate ties retain meaningful setting-specific separation or crossings, including hidden/OOD rank changes; and *discarded* when baselines collapse to noise in some setting. A canonical leaderboard requires at least one valid baseline metric; evaluation is marked complete only after a successful sweep with non-empty metric columns.

E Benchmark Scope and Relation to Prior Work

Scope and selection. CreativeBench2.0 measures component-level discovery within fixed, attributable research codebases. It does not evaluate anchor-free problem formation or unrestricted codebase exploration. Reusable source evaluations favor mature subfields with standardized protocols.

Table 18: **Construction stages and admission checks.** These are workflow requirements; checkpoint evidence and audit limits are reported in Appendix D.1.

Stage	Required evidence or exit condition
Survey	No-install screening of source, environment, evaluation entry, and at least three baselines; record proceed, patch, or stop verdict.
Scout	A reusable multi-method source, an isolable component, ready data, and at least three paper-grounded competitors sharing a metric and a paper-scale evaluation feasible within budget.
Verify	Real paper/tool records, trustworthy code links, a common component interface, and at least one paper from the last three years. Quality gate: at least 50 citations (20 for 2024–2026), or a top-venue whitelist.
Analyze	Reproduce published results before porting; preserve baseline order after porting, with multi-seed Spearman $\rho \approx 1$.
Scope	An edit surface supporting every baseline (suggested 50–200 lines). Exclude optimizer controls, data loading, architecture settings, metrics, logging, and constant-only shortcuts.
Baseline	At least three mechanism-diverse executable edits with full arXiv references; paths and line ranges match the scoped source.
Assemble	Emit the task manifest and provenance; validate edit application, Python compilation, legal operations, code-root resolution, parser/specification keys, and executable scoring.
Validate	Structural and parser checks pass, a dry-run sweep emits a leaderboard, six-dimension LLM-judge mean $\geq 3.5/5$, no shortcut, and overall PASS.
Review	Produce a review report and human checklist, record unresolved risks, then hand off to the GPU sweep.
Calibrate	Apply the post-sweep rank verdicts in Appendix D.3; discarded tasks record their failure type and recovery destination.

Both tracks inherit task-specific metrics from empirical baseline anchors; anchor errors can propagate despite the rank-fidelity gates. Human performance is currently unmeasured.

Relationship to the earlier CreativeBench. CreativeBench (Wang et al., 2026b) evaluates combinatorial and exploratory creativity through automatically constructed programming tasks and a quality–novelty score. CreativeBench2.0 shares automated construction and execution-grounded evaluation, extending the task interface to ML components, visible/hidden OOD settings, baseline-rank checks, and pre-execution Foresight. Its Discovery score measures baseline-relative performance without the earlier novelty factor; completed qualitative human review is reported in Appendix C.3. Public-source exposure and construction-cost limits appear in Appendices B.4 and D.1.