SELFAI: A SELF-DIRECTED MULTI-AGENT FRAMEWORK FOR LONG-HORIZON SCIENTIFIC DISCOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agents increasingly explore scientific problems on their own, yet they are judged almost solely by the final result they report. That result cannot distinguish an efficient investigation from one that spent most of its budget on redundant trials. We introduce SelfAI, a self-directed multi-agent system that treats experimentation as trajectory-driven decision making. Given a research objective and experimental scope, SelfAI formalizes the task, reasons over accumulated trajectories to plan subsequent trials, and decides when further trials are no longer worthwhile. To evaluate these decisions directly, we build a controlled benchmark in which all methods face the same executed experiments and observe each outcome only after selecting it. Across tasks spanning scientific computing, machine learning, computer vision, medical image analysis, and drug discovery, SelfAI finds the best configurations nearly as reliably as full-budget classical optimization methods and LLM baselines while using substantially fewer trials, whereas early-stopping baselines with similar trial counts miss them more often. Ablations show that trajectory analysis, planning, and stopping make complementary contributions. The evaluation further reveals that methods with similar final quality can differ substantially in how many trials they spend and when they stop. Code and VSCode extensions will be released.

1 INTRODUCTION

Scientific discovery across engineering (Bradshaw et al., 1983; Angelopoulos et al., 2024), physics (Lupoiu et al., 2025; Kaiser et al., 2025), biology (Lyu et al., 2024; Nguyen et al., 2024), and medicine (Yakavets et al., 2025; Grisoni et al., 2021; Jiang et al., 2022) depends on sequences of experiments. Each result can change which direction merits further investigation, how resources should be allocated, and whether additional experimentation is worthwhile. AI systems increasingly assist individual stages of this process, from predictive modeling to experiment design and execution (Xin et al., 2025). Coordinating these stages as evidence accumulates remains a central challenge.

Most existing AI-assisted discovery systems (Toledo et al., 2025; Yakavets et al., 2025; Erps et al., 2021) emphasize predictive modeling and judge success mainly by final performance. They offer limited support for reasoning about the experimental process itself: which experiment to run next, how to allocate a limited budget, and when further trials are no longer worthwhile. Final performance alone cannot reveal how well these decisions were made. A strong result may follow either an efficient investigation or many redundant attempts, and an early stop may reflect either sufficient progress or premature abandonment. This work focuses on the underlying problem: how should an agent use an investigation's history to revise its next experiment and decide whether to continue?

Large language models (LLMs) (OpenAI, 2023; Bai et al., 2023; Guo et al., 2025) have expanded what AI systems can contribute to this process by connecting reasoning, tool use, and experimental feedback (Huang & Chang, 2022; Ferrag et al., 2025). Early work showed that LLMs can extract scientific knowledge to guide experiments (Lin et al., 2023; Angelopoulos et al., 2024) and answer domain-specific questions (Alampara et al., 2025; Polak & Morgan, 2024; Dagdelen et al., 2024). Later systems extended these abilities to cross-stage planning, as in Biomni (Huang et al., 2025), to hypothesis generation (Wang et al., 2025; Toledo et al., 2025; Baek et al., 2025), and to multimodal

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

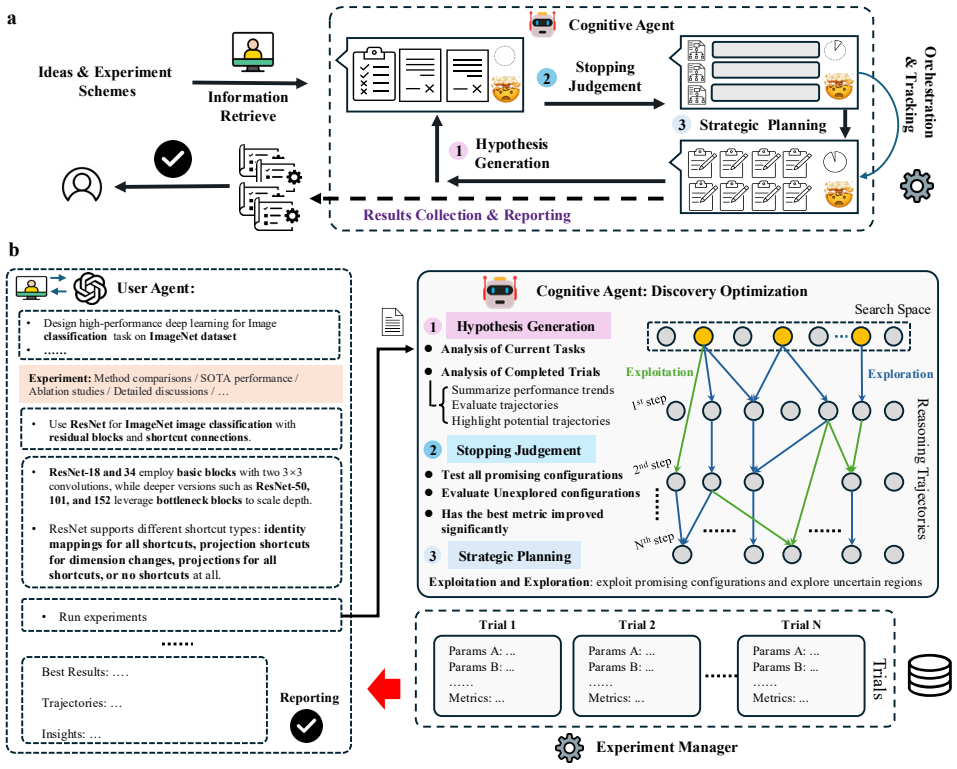

Figure 1: **SelfAI framework for iterative scientific experimentation. a**, The multi-agent architecture connects task formalization, trajectory analysis, hypothesis generation, planning, stopping judgment, and execution feedback. **b**, User ideas and experimental plans are translated into structured task specifications. The Cognitive Agent uses accumulated results to refine proposals and decide whether to continue, while Experiment Manager coordinates execution and result collection.

knowledge integration (Steyaert et al., 2023; Gao et al., 2025). Research agents such as AIDE (Jiang et al., 2025), R&D-Agent (Yang et al., 2025), AI Scientist (Lu et al., 2024; Yamada et al., 2025; Lu et al., 2026), and AgenticSciML (Jiang & Karniadakis, 2025) organize iterative research workflows end to end, building on earlier autonomous systems such as the Robot Scientist (King et al., 2004). Related agents also operate laboratory instruments (Mandal et al., 2025) and generate hypotheses collaboratively, as in AI co-scientist (Gottweis et al., 2025).

These systems demonstrate that research workflows can be automated at the system level, but they mainly improve individual stages or optimize the final outcome. LLM-driven optimization methods (Liao et al., 2025; Kochnev et al., 2025), which propose candidates or synthesize update rules, likewise operate largely at the level of local improvement. Experimentation, however, is sequential: each trial contributes both a result and evidence for later choices, so trial allocation and stopping are decisions over the whole experimental trajectory. We call such history-dependent, multi-round decisions *long-horizon*; the term refers to reliance on accumulated evidence, not to unbounded search spaces or research programs of arbitrary length.

We introduce SelfAI, a self-directed multi-agent framework that treats long-horizon experimentation as a trajectory-level decision process. Starting from a user-specified objective and experimental scope, SelfAI uses the accumulating record to guide subsequent trials within those boundaries (Fig. 1). A **User Agent** translates research intent into a structured task specification. A **Cognitive Agent** analyzes completed trials, generates hypotheses about which experiments to run, plans subsequent trials, and decides whether further experimentation is worthwhile. An **Experiment Manager** coordinates execution and records feedback. The shared history connects these roles throughout the investigation.

Evaluating this process presents a methodological challenge. In end-to-end agent workflows, outcomes depend jointly on candidate generation, implementation, execution, planning, and stop-

ping (Huang et al., 2024; Chan et al., 2024; Toledo et al., 2025), so the final score cannot isolate how well an agent uses accumulated evidence. We therefore build a controlled benchmark that fixes the candidate experiments and their completed outcomes across methods and reveals each outcome only after its configuration is selected. Every method follows its own adaptive path through the same experimental landscape, which makes trial allocation and stopping directly comparable and their trajectories inspectable.

To characterize trajectories, we introduce two complementary metrics. Score combines normalized endpoint quality with the time to reach the best reference result and the time of stopping. AUP_D evaluates the performance diversity of observed outcomes: it grows with the number of trials and with how broadly their performance extends across the reference range, capturing the efficiency–diversity trade-off of a trajectory. Because it is computed on outcomes and is order-invariant, it is interpreted together with endpoint quality, whether and when the best reference result is found, and when search stops, to distinguish productive exploration from redundant trials or premature stopping.

Our contributions are:

1. **Scientific discovery as a trajectory-level decision process.** We frame scientific discovery as an exploration process in which hypothesis generation, trial allocation, and stopping are guided by the accumulated experimental history under a limited budget, and implement it in SelfAI through a User Agent, a Cognitive Agent, and an Experiment Manager.
2. **Evaluation of experimental trajectories.** A controlled benchmark and two trajectory metrics, Score and AUP_D, reported alongside endpoint metrics, characterize the efficiency–diversity trade-off of observed outcomes, when useful results are reached, and when experimentation ends.
3. **Validation across scientific domains.** Across tasks from scientific computing, machine learning, computer vision, medical image analysis, and drug discovery, SelfAI finds the best configurations with substantially fewer trials than baselines of comparable hit rate. Ablations show that trajectory analysis, planning, and stopping make complementary contributions, and the controlled evaluation reveals that methods with similar final quality can differ substantially in how many trials they spend and when they stop.

2 Related Works

Here, we review three lines of related work: agent-centric benchmarks and pipelines, scientific discovery systems and benchmarks, and sequential and language-model optimization.

Agent-Centric Benchmarks and Pipelines Self-evolution frameworks such as SE-Agent (Lin et al., 2025), AIRA (Toledo et al., 2025), and EvoX (Liu et al., 2026) refine agent behavior through evolution or search operators, but they typically evaluate large numbers of candidates and are judged by the quality of what they finally produce. Their evaluation metrics follow the same endpoint view. pass@k (Chen et al., 2021) estimates the probability that at least one of k independent samples succeeds. HeuriGym's Quality–Yield (Chen et al., 2025) complements it by measuring the fraction of feasible solutions and their quality relative to expert performance. MLGym's performance profiles and AUP (Nathani et al., 2025) aggregate each method's final score relative to the best method across tasks. These metrics remain centered on final performance and candidate feasibility. They do not record when a strong configuration was first found, how many trials followed it, or when the overall search stopped. In contrast, SelfAI evaluates the observable trial sequence of the exploration process, including when the best configuration is discovered and when the overall search stops (Sect. 3.2).

Scientific Discovery Systems and Benchmarks MLAgentBench (Huang et al., 2024) evaluates agents that edit files, execute code, and inspect results across machine-learning research tasks. MLE-Bench (Chan et al., 2024) evaluates end-to-end engineering on Kaggle competitions. AIDE (Jiang et al., 2025) represents candidate programs as a solution tree and applies draft, debug, and improve operators. R&D-Agent (Yang et al., 2025) schedules research traces that couple hypotheses with implementation and execution. AI Scientist (Lu et al., 2024; Yamada et al., 2025; Lu et al., 2026) and NovelSeek (Team et al., 2025) cover broader paper- or hypothesis-oriented workflows. These systems already maintain and use experimental histories. However, their end-to-end outcome scores jointly

reflect code quality and correctness, execution reliability, validation fidelity, and search decisions, so they do not isolate the quality of exploration or discovery. SelfAI connects explicit trajectory analysis, planning, and stopping with task formalization and execution feedback. Our open-ended evaluation on MLE-Bench examines this evaluation confounding, while the controlled evaluation measures the contribution of these decision functions under shared conditions.

Sequential and Language-Model Optimization Optuna's Tree-structured Parzen Estimator (TPE) (Akiba et al., 2019), SMAC, Random Search, and Successive Halving are representative policies for allocating trials in bounded configuration spaces. MCTS (Świechowski et al., 2023) adds explicit tree selection, while Optimization by PROmpting (OPRO) (Yang et al., 2023) asks an LLM to propose solutions from previously evaluated solution–value pairs. These methods motivate our matched baselines. Learning-curve methods such as Hyperband (Li et al., 2018), ASHA (Li et al., 2020), BOHB (Falkner et al., 2018), and freeze–thaw optimization (Swersky et al., 2014) address within-training resource allocation. Our benchmark exposes completed-trial outcomes rather than intermediate learning curves, so we compare completed-trial anytime and overall-search stopping variants instead. Table 1 summarizes the candidate space, use of history, primary evaluation, and stopping protocol of representative methods.

Table 1: Comparison of representative optimization methods, research agents, and benchmarks. Entries describe their native protocols. They are not broad capability judgments.

Method or benchmark	Candidate/action space	Use of history	Primary evaluation	Overall-search stopping
Optuna/TPE (Akiba et al., 2019)	fixed configurations	density-based proposal	best objective	optional/user-defined
OPRO (Yang et al., 2023)	generated text solutions	solution–value prompt	best objective	budget/criterion
MLAgentBench (Huang et al., 2024)	code and shell actions	ReAct interaction trace	task success	budget-limited
MLE-Bench (Chan et al., 2024)	open-ended ML code	agent-dependent	private-test score/medal	time budget
AIDE (Jiang et al., 2025)	generated program tree	node selection and repair	validation/private score	search budget
R&D-Agent (Yang et al., 2025)	hypotheses, code, research traces	scheduled trace evolution	solution performance	workflow-dependent
SelfAI (controlled)	fixed unevaluated configurations	trial analysis and planning	trajectory and endpoint metrics	explicit stop judgment

3 METHODS

3.1 SELFAI SYSTEM

SelfAI is a self-directed multi-agent framework for iterative scientific experimentation. As shown in Fig. 1, it connects task formalization, trajectory analysis, hypothesis generation, experiment planning, execution feedback, and stopping judgment in a stateful loop. The User Agent translates research intent into a structured specification. The Cognitive Agent uses the history of completed experiments to revise proposals and decide whether to continue. The Experiment Manager coordinates execution and returns observations to the shared history. This organization enables later decisions to incorporate evidence from earlier rounds within the user's experimental scope.

User Agent As shown in Fig. 1b, the User Agent converts natural-language goals, experimental ideas, constraints, and prior observations into a task-specific YAML specification (Supplementary Sect. A.2). The specification records the objective, admissible experimental choices, evaluation criteria, and budget required by the task. Users can inspect and revise it before execution. During a run, SelfAI selects trials and decides when to stop without trial-level user input, while users can still revise the objective, constraints, or candidate space by updating the specification. The specification thus provides a consistent interface for the Cognitive Agent and Experiment Manager throughout the experiment–feedback loop.

Cognitive Agent The Cognitive Agent (Fig. 1b, upper right) receives the task specification and ordered experimental history and applies three connected functions. **Trajectory analysis and hypothesis generation** summarizes observed outcomes, failures, and unresolved directions, and generates hypotheses, that is, candidate experiments expected to improve the objective or to be informative. **Stopping judgment** assesses recent progress, remaining opportunities, and available resources to decide whether further trials are warranted. If the search continues, **strategic planning** selects the experiments to run from these hypotheses, balancing promising directions with further exploration. In the controlled benchmark, each hypothesis is a not-yet-evaluated configuration from

the task's finite candidate space: trajectory analysis identifies which of these configurations are promising or informative, strategic planning selects the ones to run, and the Experiment Manager returns the outcome of each selected configuration, which was executed in advance under fixed settings. This restriction is an evaluation choice that isolates selection and stopping. In the open-ended evaluation on MLE-Bench, SelfAI selects draft, debug, or improve actions and judges stopping, while the native agents generate and execute new code candidates. That evaluation is not an attribution test: endpoint results jointly reflect code quality, execution reliability, validation fidelity, selection, and stopping. Prompt templates are given in Supplementary Sect. A.3.

Experiment Manager The Experiment Manager (Fig. 1b, lower right) coordinates the execution and management of experiments proposed by the Cognitive Agent. Its responsibilities include environment preparation, resource coordination, task launch, progress tracking, checkpoint-based recovery, failure reporting, and structured result logging. Completed outcomes and execution states are appended to the shared history and returned to the Cognitive Agent as evidence for subsequent reasoning. In the controlled benchmark, each configuration has already been executed under fixed settings, so the Experiment Manager returns that execution result directly. Every method that selects the configuration therefore receives the same outcome, and training need not be repeated. In open workflows, the proposed experiment is executed through the available tools and computing environment. The module provides the infrastructure required to close the reasoning–execution–feedback loop and is not presented as a new scheduling algorithm.

3.2 Evaluation for reasoning trajectories

Endpoint measures such as pass@k or the best final objective do not reveal when a useful configuration was first observed or how many trials followed it. We therefore evaluate the observable sequence of selected configurations and returned objective values. The metrics below quantify endpoint quality, discovery time, stopping time, and the distribution of performance along that sequence. They do not score private chain-of-thought or claim to measure general reasoning ability.

Trajectory Representation The following definitions are applied separately to each compared method and run. For task i, let H_i be the common evaluation horizon, $o_i \in \{\text{ascend}, \text{descend}\}$ the metric direction, and

$$\mathcal{T}_i = ((x_{i,k}, v_{i,k}))_{k=1}^{n_i}, \qquad n_i \le H_i,$$

the observed search trajectory. The metrics require a shared evaluation reference $\mathcal{R}_i$. In the controlled benchmark, $\mathcal{R}_i$ is the complete outcome table, and it is updated after all compared methods have run by pooling their recorded candidate spaces: $\mathcal{R}_i = \bigcup_{a \in \mathcal{A}} \mathcal{X}_{i,a}$, where $\mathcal{A}$ is the set of compared methods and $\mathcal{X}_{i,a}$ is the candidate space recorded for method a. Each method is then evaluated from its own trajectory against this same post-hoc reference. Pooling makes the within-comparison normalization symmetric: no method is evaluated against a reference unavailable to the others.

Let v_i^* be the best observed value and v_i^{ref} the best value in $\mathcal{R}_i$. If the trajectory reaches v_i^{ref}, m_i is its first-hit index. Otherwise, $m_i = H_i$. With $v_{i,\min}$, $v_{i,\max}$, and $\Delta_i = v_{i,\max} - v_{i,\min}$ computed from $\mathcal{R}_i$,

$$\text{Gain}_i = \begin{cases} (v_i^* - v_{i,\min})/\Delta_i, & o_i = \text{ascend},\ \Delta_i > 0, \\ (v_{i,\max} - v_i^*)/\Delta_i, & o_i = \text{descend},\ \Delta_i > 0, \\ 1, & \Delta_i = 0, \end{cases} \tag{1}$$

$$t_i^{\text{best}} = m_i/H_i, \qquad t_i^{\text{stop}} = n_i/H_i, \tag{2}$$

$$\text{Score} = \frac{1}{N} \sum_{i=1}^{N} \text{Gain}_i \left(1 - \frac{t_i^{\text{best}} + t_i^{\text{stop}}}{2}\right). \tag{3}$$

A miss has $t_i^{\text{best}} = 1$. In the controlled benchmark, $H_i = |\mathcal{R}_i|$, which recovers the reported values. Metrics are computed per run, averaged across seeds within task, and then averaged across tasks. Best Norm, Hit Rate, and Trials follow the same ordering.

Performance Diversity The statistic below evaluates the performance diversity of a trajectory, that is, the distribution of its observed objective values. Its area term grows with the number of trials and

with how broadly their performance extends across the shared threshold range, and its shape term summarizes that distribution relative to the baseline. AUP_D therefore reflects the efficiency–diversity trade-off of the observed outcomes; after normalization to the baseline, lower values indicate that fewer trials are spent for the observed performance distribution. It is computed on outcomes rather than configurations and is invariant to permutations of the same observed values. Temporal behavior is represented separately by t_i^{best}, t_i^{stop}, and Score, and AUP_D is interpreted jointly with those quantities. For positive finite objectives, let $[\tau_{i,\min}, \tau_{i,\max}]$ be the log-value range of $\mathcal{R}_i$ and

$$\rho_i(\tau) = \begin{cases} |\{k : v_{i,k} \geq e^{\tau}\}|, & o_i = \text{ascend}, \\ |\{k : v_{i,k} \leq e^{\tau}\}|, & o_i = \text{descend}. \end{cases}$$

With $A_i = \int \rho_i(\tau)d\tau$, $A = N^{-1}\sum_i A_i$, $G_i = A_i^{-1}\int \tau\rho_i(\tau)d\tau$, and

$$S_i = \int \left(\frac{\tau}{G_i}\right)^3 \rho_i(\tau)d\tau, \quad S_i' = \frac{1}{2}\left[\tanh\left(1 - \frac{S_i - S_{i,\mathrm{base}}}{S_{i,\mathrm{base}}}\right) + 1\right],$$

we define

$$\widetilde{\mathrm{AUP}}_D = \frac{A}{N^{-1}\sum_i S_i'}, \qquad \mathrm{AUP}_D = \frac{\widetilde{\mathrm{AUP}}_D}{\widetilde{\mathrm{AUP}}_{D,\mathrm{base}}}.$$

The baseline uses the same $\mathcal{R}_i$, horizon, and thresholds. We use GS in the controlled benchmark. Numerical and missing-value rules appear in Supplementary Sect. A.1.

4 Experiment

We evaluate how SelfAI uses accumulated evidence to allocate trials and decide when to stop. The controlled benchmark contains 12 task settings from scientific computing, machine learning, computer vision, medical image analysis, and drug discovery (Supplementary Table 6). Many are hyperparameter or model-configuration tasks, which provide reproducible experimental choices for this comparison. Each YAML instance specifies the objective, metric direction, finite candidate space, initial completed trials, and unique-trial budget. Most outcome tables were produced offline by systematic execution; LCBench (Zimmer et al., 2021) and Chagas EP20 originate from Bayesian-optimization trajectories. Solvers observe an outcome only after selecting its configuration and choose their own sequence of trials. Fixing the underlying experiments makes differences in trial allocation and stopping comparable across methods.

All methods share the same candidate space, initial trials, observation-history format, unique-trial budget, and YAML output schema. We compare Grid Search (GS), TPE Bayesian Search (BS) (Akiba et al., 2019), SMAC3 (Lindauer et al., 2022), Random Search (Bergstra & Bengio, 2012), Successive Halving (SHA) (Jamieson & Talwalkar, 2016), MCTS (Świechowski et al., 2023), OPRO (Yang et al., 2023), LLM, LLM-MCTS, their online-stopping variants, and SelfAI variants; all methods are defined in Supplementary Sect. B.1. **LLM** directly generates experimental configurations from the task description without iterative feedback. **LLM-ES** receives the same completed and unexplored trials and the same stop-or-continue action space as SelfAI, but not SelfAI's trajectory-analysis and strategic-planning context. The prompts for LLM and LLM-ES are provided in Supplementary Sect. H. ASHA, BOHB, Hyperband, and freeze–thaw methods require intermediate learning curves, which are unavailable in this completed-trial benchmark. Their role is instead approximated by anytime and overall-search stopping baselines. All LLM-based methods (LLM, LLM-ES, OPRO, the LLM-MCTS variants, and SelfAI) use the same GPT4-o3-mini backbone for the principal comparison and receive the same task specification, including the task description and any provided domain knowledge. We further instantiate SelfAI with GPT4-o3 (OpenAI, 2023), Llama3.3-70b (Touvron et al., 2023), Qwen2.5-7b/14b/32b/72b (Bai et al., 2023), and DeepSeek-R1-7b/14b/32b/70b (Guo et al., 2025) (Table 2), where DeepSeek-R1-7b/14b/32b/70b denote the corresponding DeepSeek-R1-Distill models. All methods use identical computational settings: random seeds are fixed, temperature is set to zero, and reasoning traces are retained for analysis and reproducibility.

4.1 Benchmark results

Score combines normalized endpoint quality with penalties for delayed discovery of the shared-reference best and late stopping. In the controlled benchmark, the shared reference is the complete

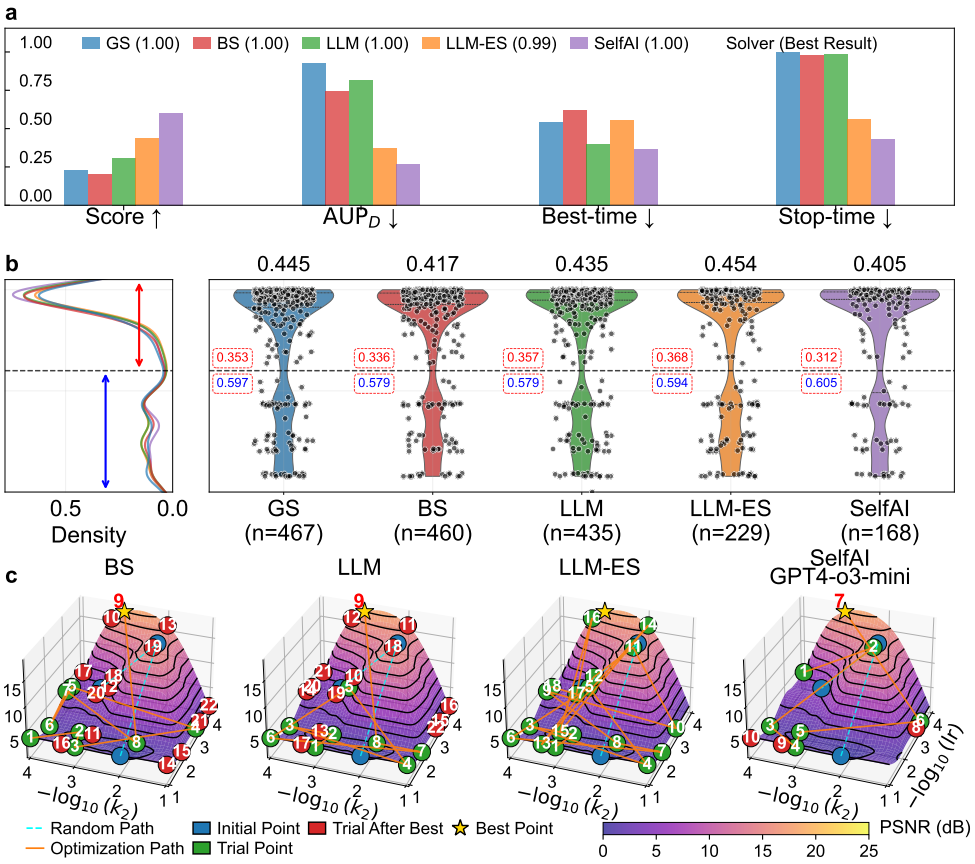

Figure 2: **Performance comparison and efficiency-diversity trade-offs across solvers. a**, Normalized performance of five solvers. **b**, Outcome distributions across solvers. Left: kernel density estimates illustrating score concentration and convergence tendencies. Right: violin plots with overlaid individual trials (sample size n shown), capturing performance variability. The dashed horizontal line denotes the evaluation threshold separating low- and high-performance regimes; annotated statistics (red/blue) highlight solver behavior above and below this threshold, where broader exploration facilitates rapid escape from low-performance regions and tighter distributions in high-performance regions enable localized refinement. **c**, Illustration of the optimized trajectory for the SIREN method for image segmentation. Green points are suggested points before reaching the optimal points. Red points are redundant suggestions when reaching out to the optimal points and failing to stop trials. The $\star$ is the optimal point. SelfAI demonstrates more structured exploration and faster convergence towards high-performance regions.

finite outcome table (Sect. 3.2). A miss is assigned $t_{\text{best}} = 1$. LCBench is excluded from the aggregate. Stochastic methods are run with multiple random seeds. Metrics are first averaged across runs within each problem and then equally across problems. Table 3 and Fig. 2a show the quality–cost trade-off. SelfAI achieves the highest Score (0.6019), the earliest mean reference-best discovery time ($t_{\text{best}} = 0.3642$), and a 0.9231 Hit Rate with 168 runs. It also outperforms every stopping-enabled baseline in Score and uses the fewest trials among methods with comparable Hit Rate. AUP$_D$ evaluates the performance diversity of observed outcomes and reflects their efficiency–diversity trade-off (Fig. 2b); because it is order-invariant, timing comparisons use t_{best}, t_{stop}, and Score.

4.2 Trajectory patterns in the controlled benchmark

Across the evaluated landscapes, successful trajectories commonly move from initial sampling to refinement of a promising region and then stop after marginal gains diminish. Figure 2c illustrates this pattern for SIREN.

Table 2: Averaged performance comparison of SelfAI across different tasks. The problems are those of Table 3 plus LCBench, on which GPT4-o3-mini and GPT4-o3 are not run (Supplementary Sect. B); their metrics, including Hit Rate, are averaged over the remaining problems. Rank orders the solvers by Score.

Solver	Score↑	AUP$_D$ ↓	t_{best} ↓	t_{stop} ↓	Hit Rate↑	Rank
Llama3.3-70b	0.3786	0.4983	0.5182	0.6654	0.7143	11
Qwen2.5-7b	0.5701	0.2324	0.4476	0.4015	0.7857	2
Qwen2.5-14b	0.4923	0.3715	0.4807	0.5306	0.7857	6
Qwen2.5-32b	0.4356	0.4739	0.5168	0.6084	0.7857	10
Qwen2.5-72b	0.4426	0.5086	0.4436	0.6708	0.8571	9
DeepSeek-R1-7b	0.4965	0.2072	0.6455	0.3317	0.5000	5
DeepSeek-R1-14b	0.4912	0.3845	0.4814	0.5053	0.7143	7
DeepSeek-R1-32b	0.5010	0.3213	0.4748	0.5160	0.7857	4
DeepSeek-R1-70b	0.4660	0.3520	0.5255	0.5256	0.7143	8
GPT4-o3-mini	0.6019	0.2647	0.3642	0.4313	0.9231	1
GPT4-o3	0.5092	0.2818	0.5159	0.4418	0.6923	3

Backbone-specific results (Table 2) do not show monotonic improvement with model size in this benchmark. Larger models often sustain prolonged exploration of alternative hypotheses, leading to delayed commitment and reduced adaptability under fixed experimental budgets. In contrast, small and mid-sized models often exhibit more stable cumulative progress, earlier concentration on informative regions, and earlier stopping. More importantly, comparable endpoint quality can coexist with different trajectory costs: full-budget methods attain Best Norm near 1 but pay maximal stop-time penalties, whereas aggressive stopping methods save trials but frequently miss high-performing candidates. The paired endpoint and process columns in Tables 3–4 make this distinction explicit.

Ablation study. Table 4 shows that SelfAI's gains are not due to stopping timing alone. Removing trajectory analysis barely changes the stop time but lowers Score and Hit Rate and delays discovery of the best configuration; LLM-ES, which removes both analysis and planning, likewise lowers Score and Hit Rate. Removing strategic planning stops earlier and lowers AUP$_D$, but the search often ends before reaching the best configuration, which lowers Hit Rate and Score and raises t_{best}. Patience stopping prolongs the search and lowers Score, while disabling stopping reaches the best endpoint quality only by exhausting the budget, using nearly three times as many trials for a marginal gain in Best Norm. Replacing SelfAI's search with LLM-MCTS under the same stopping judgment also lowers Score markedly (Supplementary Table 5). Overall, SelfAI couples early discovery, moderate stop time, and high Hit Rate with the best Score, and the three components are complementary.

On MLE-Bench (Chan et al., 2024), SelfAI helps in some settings but not in others, because code quality, execution reliability, and validation fidelity jointly determine the endpoint (Supplementary Sect. B.7); the controlled benchmark fixes these factors so that only selection and stopping differ.

Limitations This work has several limitations, including dependence on the underlying language model, assumptions on structured search spaces, and computational overhead in large-scale settings. A detailed discussion is provided in Supplementary Sect. D.

5 Conclusion

In this paper, we present SelfAI, a self-directed multi-agent framework for long-horizon scientific discovery in which a User Agent formalizes research intent, a Cognitive Agent analyzes accumulated trials to generate hypotheses, judge when to stop, and plan subsequent experiments, and an Experiment Manager executes them. SelfAI further introduces a process-aware evaluation perspective: Score

Table 3: Averages over the controlled benchmark tasks, excluding LCBench. For stochastic methods, per-run metrics are averaged over random seeds within problem before equal cross-problem averaging. ES denotes online overall-search stopping.

Method	Score↑	AUP_D ↓	t_{best} ↓	t_{stop} ↓	#Trials↓	Best Norm↑	Hit Rate↑
Without online overall-search stopping							
GS	0.2330	1.0000	0.5341	1.0000	35.92	1.0000	1.0000
BS (TPE)	0.2163	0.7976	0.5889	0.9785	35.38	1.0000	1.0000
SMAC3	0.3759	0.6684	0.3953	0.8528	29.20	0.9999	0.9692
Random	0.2837	0.8067	0.4327	1.0000	35.92	1.0000	0.9846
SHA	0.2053	0.7770	0.5894	1.0000	35.92	1.0000	0.9231
MCTS	0.2651	0.7770	0.4697	1.0000	35.92	1.0000	0.9231
LLM	0.3311	0.7602	0.3987	0.9390	26.23	1.0000	1.0000
LLM-MCTS	0.2113	0.7770	0.5774	1.0000	35.92	1.0000	0.9231
OPRO	0.2722	0.7770	0.4557	1.0000	35.92	1.0000	0.9231
SelfAI w/o stopping	0.2899	0.8100	0.4202	1.0000	35.92	1.0000	1.0000
With online overall-search stopping							
Random-ES	0.4387	0.2332	0.6753	0.3552	9.48	0.9396	0.4308
BS-ES (TPE-ES)	0.4526	0.2067	0.6670	**0.3342**	**8.92**	0.9310	0.3846
MCTS-ES	0.4401	0.2322	0.6137	0.4043	10.31	0.9016	0.5385
MCTS-UCT-stop	0.3399	0.6699	0.4288	0.8914	32.69	1.0000	1.0000
LLM-ES	0.4370	0.3886	0.5529	0.5622	17.62	0.9903	0.6923
Rule-SelfAI	0.4164	0.5350	0.4456	0.7205	27.15	0.9994	0.7692
LLM-MCTS-ES	0.3571	**0.1724**	0.7867	0.3411	9.38	0.8557	0.3077
SelfAI	**0.6019**	0.2647	**0.3642**	0.4313	12.92	0.9981	0.9231

Table 4: Component ablations of SelfAI. TA denotes trajectory analysis and hypothesis generation. SP denotes strategic planning. SJ denotes SelfAI stopping judgment. PS denotes patience stopping.

Method	TA	SP	SJ	PS	Score↑	AUP_D ↓	t_{best} ↓	t_{stop} ↓	Best Norm↑	Hit Rate↑	#Trials↓
SelfAI	✓	✓	✓	✗	**0.6019**	0.2647	**0.3642**	0.4313	0.9981	0.9231	**12.92**
SelfAI w/o trajectory analysis	✗	✓	✓	✗	0.4466	0.2970	0.6714	0.4261	0.9867	0.5385	16.08
SelfAI w/o strategic planning	✓	✗	✓	✗	0.5138	**0.1952**	0.6137	**0.3454**	0.9864	0.5385	13.38
SelfAI w/ patience stopping	✓	✓	✗	✓	0.4164	0.5350	0.4456	0.7205	0.9994	0.7692	27.15
SelfAI w/o stopping	✓	✓	✗	✗	0.2899	0.8100	0.4202	1.0000	1.0000	1.0000	35.92
LLM-ES	✗	✗	✓	✗	0.4370	0.3886	0.5529	0.5622	0.9903	0.6923	17.62
LLM	✗	✗	✗	✗	0.3311	0.7602	0.3987	0.9390	1.0000	1.0000	26.23

captures when useful results are reached and when experimentation ends, and AUP_D captures how efficiently trials are spent. Under this evaluation, SelfAI reaches the best configuration with substantially fewer trials than baselines of comparable hit rate, and methods with similar final quality differ markedly in how many trials they spend.

AI USE STATEMENT

Generative AI systems are both the subject of this study and components of the evaluated agent workflows. They were used to produce the model-driven selections and stopping decisions analyzed in the experiments and to assist with manuscript language and LaTeX formatting. The authors reviewed the experimental records, numerical tables, citations, claims, and AI-assisted manuscript revisions, and take responsibility for the final content.

REPRODUCIBILITY STATEMENT

Section 3.1 defines the SelfAI system workflow and the responsibilities of its three modules, Sect. 3.2 defines the trajectory metrics, and Sect. 4 specifies the shared observation and budget settings

used in the controlled evaluation. The Supplementary Materials provide the YAML interface, complete prompt templates, task-level settings, and additional results needed to reproduce the reported comparisons. Code and machine-readable benchmark artifacts will be released anonymously with the submission.

REFERENCES

Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630(8016):493–500, 2024.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631, 2019.

Nawaf Alampara, Mara Schilling-Wilhelmi, Martiño Ríos-García, Indrajeet Mandal, Pranav Khetarpal, Hargun Singh Grover, NM Anoop Krishnan, and Kevin Maik Jablonka. Probing the limitations of multimodal language models for chemistry and materials research. *Nature computational science*, pp. 1–10, 2025.

Angelos Angelopoulos, James F Cahoon, and Ron Alterovitz. Transforming science labs into automated factories of discovery. *Science Robotics*, 9(95):eadm6991, 2024.

Jinheon Baek, Sujay Kumar Jauhar, Silviu Cucerzan, and Sung Ju Hwang. Researchagent: Iterative research idea generation over scientific literature with large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6709–6738, 2025.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Aleksandr Berezutskii, Minzhao Liu, Atithi Acharya, Roman Ellerbrock, Johnnie Gray, Reza Haghshenas, Zichang He, Abid Khan, Viacheslav Kuzmin, Dmitry Lyakh, et al. Tensor networks for quantum computing. *Nature Reviews Physics*, pp. 1–13, 2025.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

Pietro Bongini, Monica Bianchini, and Franco Scarselli. Molecular generative graph neural networks for drug discovery. *Neurocomputing*, 450:242–252, 2021.

Gary F Bradshaw, Patrick W Langley, and Herbert A Simon. Studying scientific discovery by computer simulation. *Science*, 222(4627):971–975, 1983.

Shreyas Bhat Brahmavar, Ashwin Srinivasan, Tirtharaj Dash, Sowmya Ramaswamy Krishnan, Lovekesh Vig, Arijit Roy, and Raviprasad Aduri. Generating novel leads for drug discovery using llms with logical feedback. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 21–29, 2024.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, et al. Mle-bench: Evaluating machine learning agents on machine learning engineering. *arXiv preprint arXiv:2410.07095*, 2024.

Hongzheng Chen, Yingheng Wang, Yaohui Cai, Hins Hu, Jiajie Li, Shirley Huang, Chenhui Deng, Rongjian Liang, Shufeng Kong, Haoxing Ren, et al. Heurigym: An agentic benchmark for llm-crafted heuristics in combinatorial optimization. *arXiv preprint arXiv:2506.07972*, 2025.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

John Dagdelen, Alexander Dunn, Sanghoon Lee, Nicholas Walker, Andrew S Rosen, Gerbrand Ceder, Kristin A Persson, and Anubhav Jain. Structured information extraction from scientific text with large language models. *Nature communications*, 15(1):1418, 2024.

Sean Ekins, Jair Lage de Siqueira-Neto, Laura-Isobel McCall, Malabika Sarker, Maneesh Yadav, Elizabeth L Ponder, E Adam Kallel, Danielle Kellar, Steven Chen, Michelle Arkin, et al. Machine learning models and pathway genome data base for trypanosoma cruzi drug discovery. *PLoS neglected tropical diseases*, 9(6):e0003878, 2015.

Timothy Erps, Michael Foshey, Mina Konaković Luković, Wan Shou, Hanns Hagen Goetzke, Herve Dietsch, Klaus Stoll, Bernhard von Vacano, and Wojciech Matusik. Accelerated discovery of 3d printing materials using data-driven multiobjective optimization. *Science advances*, 7(42): eabf7435, 2021.

Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *International conference on machine learning*, pp. 1437–1446. PMLR, 2018.

Mohamed Amine Ferrag, Norbert Tihanyi, and Merouane Debbah. From llm reasoning to autonomous ai agents: A comprehensive review. *arXiv preprint arXiv:2504.19678*, 2025.

Anton F Fliri, William T Loging, Peter F Thadeio, and Robert A Volkmann. Biological spectra analysis: linking biological activity profiles to molecular structure. *Proceedings of the National Academy of Sciences*, 102(2):261–266, 2005.

Fan Gao, Hongqiang Li, Zhilong Chen, Yunai Yi, Shihao Nie, Zihao Cheng, Zeming Liu, Yuanfang Guo, Shumin Liu, Qizhen Qin, et al. A chemical autonomous robotic platform for end-to-end synthesis of nanoparticles. *Nature Communications*, 16(1):7558, 2025.

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

Francesca Grisoni, Berend JH Huisman, Alexander L Button, Michael Moret, Kenneth Atz, Daniel Merk, and Gisbert Schneider. Combining generative artificial intelligence and on-chip synthesis for de novo drug design. *Science Advances*, 7(24):eabg3338, 2021.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Wolfgang Hackbusch. Numerical tensor calculus. *Acta numerica*, 23:651–742, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

Brad Huang. [03/24] boston housing dataset. https://kaggle.com/competitions/2403-boston-housing-dataset, 2020. Kaggle.

Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*, 2022.

Kexin Huang, Serena Zhang, Hanchen Wang, Yuanhao Qu, Yingzhou Lu, Yusuf Roohani, Ryan Li, Lin Qiu, Junze Zhang, Yin Di, et al. Biomni: A general-purpose biomedical ai agent. *bioRxiv*, pp. 2025–05, 2025.

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, 2024.

Fabian Isensee, Paul F Jaeger, Simon AA Kohl, Jens Petersen, and Klaus H Maier-Hein. nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature methods*, 18(2):203–211, 2021.

Fabian Isensee, Tassilo Wald, Constantin Ulrich, Michael Baumgartner, Saikat Roy, Klaus Maier-Hein, and Paul F Jaeger. nnu-net revisited: A call for rigorous validation in 3d medical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 488–498. Springer, 2024.

Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pp. 240–248, 2016.

Qile Jiang and George Karniadakis. Agenticsciml: Collaborative multi-agent systems for emergent discovery in scientific machine learning. *arXiv preprint arXiv:2511.07262*, 2025.

Yibin Jiang, Daniel Salley, Abhishek Sharma, Graham Keenan, Margaret Mullin, and Leroy Cronin. An artificial intelligence enabled chemical synthesis robot for exploration and optimization of nanomaterials. *Science advances*, 8(40):eabo2626, 2022.

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

Jan Kaiser, Anne Lauscher, and Annika Eichler. Large language models for human-machine collaborative particle accelerator tuning through natural language. *Science advances*, 11(1):eadr4173, 2025.

Irfan Ali Kandhro, Sahar Zafar Jumani, Fayyaz Ali, Zubair Uddin Shaikh, Muhammad Arshad Arain, and Aftab Ahmed Shaikh. Performance analysis of hyperparameters on a sentiment analysis model. *Engineering, Technology & Applied Science Research*, 10(4):6016–6020, 2020.

Ross D King, Kenneth E Whelan, Ffion M Jones, Philip GK Reiser, Christopher H Bryant, Stephen H Muggleton, Douglas B Kell, and Stephen G Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971):247–252, 2004.

Roman Kochnev, Arash Torabi Goodarzi, Zofia Antonina Bentyn, Dmitry Ignatov, and Radu Timofte. Optuna vs code llama: Are llms a new paradigm for hyperparameter tuning? *arXiv preprint arXiv:2504.06006*, 2025.

Alexandru Korotcov, Valery Tkachenko, Daniel P Russo, and Sean Ekins. Comparison of deep learning with multiple machine learning methods and metrics using diverse drug discovery data sets. *Molecular pharmaceutics*, 14(12):4462–4475, 2017.

Dominic LaBella, Maruf Adewole, Michelle Alonso-Basanta, Talissa Altes, Syed Muhammad Anwar, Ujjwal Baid, Timothy Bergquist, Radhika Bhalerao, Sully Chen, Verena Chung, Gian-Marco Conte, Farouk Dako, James Eddy, Ivan Ezhov, Devon Godfrey, Fathi Hilal, Ariana Familiar, Keyvan Farahani, Juan Eugenio Iglesias, Zhifan Jiang, Elaine Johanson, Anahita Fathi Kazerooni, Collin Kent, John Kirkpatrick, Florian Kofler, Koen Van Leemput, Hongwei Bran Li, Xinyang Liu, Aria Mahtabfar, Shan McBurney-Lin, Ryan McLean, Zeke Meier, Ahmed W Moawad, John Mongan, Pierre Nedelec, Maxence Pajot, Marie Piraud, Arif Rashid, Zachary Reitman, Russell Takeshi Shinohara, Yury Velichko, Chunhao Wang, Pranav Warman, Walter Wiggins, Mariam Aboian, Jake Albrecht, Udunna Anazodo, Spyridon Bakas, Adam Flanders, Anastasia Janas, Goldey Khanna, Marius George Linguraru, Bjoern Menze, Ayman Nada, Andreas M Rauschecker, Jeff Rudie, Nourel Hoda Tahon, Javier Villanueva-Meyer, Benedikt Wiestler, and Evan Calabrese. The asnr-miccai brain tumor segmentation (brats) challenge 2023: Intracranial meningioma, 2023.

Bennett Landman, Zhoubing Xu, J Igelsias, Martin Styner, T Langerak, and Arno Klein. Miccai multi-atlas labeling beyond the cranial vault–workshop and challenge. In *Proc. MICCAI Multi-Atlas Labeling Beyond Cranial Vault—Workshop Challenge*, volume 5, pp. 12, 2015.

Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-Tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. A system for massively parallel hyperparameter tuning. *Proceedings of machine learning and systems*, 2:230–246, 2020.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of machine learning research*, 18(185):1–52, 2018.

Ziguang Li, Chao Huang, Xuliang Wang, Haibo Hu, Cole Wyeth, Dongbo Bu, Quan Yu, Wen Gao, Xingwu Liu, and Ming Li. Lossless data compression by large models. *Nature Machine Intelligence*, pp. 1–6, 2025.

Rongjie Liao, Junhao Qiu, Xin Chen, and Xiaoping Li. Llm4eo: Large language model for evolutionary optimization in flexible job shop scheduling. *arXiv preprint arXiv:2511.16485*, 2025.

Jiaye Lin, Yifu Guo, Yuzhen Han, Sen Hu, Ziyi Ni, Licheng Wang, Mingguang Chen, Hongzhang Liu, Ronghao Chen, Yangfan He, et al. Se-agent: Self-evolution trajectory optimization in multi-step reasoning with llm-based agents. *arXiv preprint arXiv:2508.02085*, 2025.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. SMAC3: A versatile bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23(54):1–9, 2022.

Shu Liu, Shubham Agarwal, Monishwaran Maheswaran, Mert Cemri, Zhifei Li, Qiuyang Mang, Ashwin Naren, Ethan Boneh, Audrey Cheng, Melissa Z Pan, et al. Evox: Meta-evolution for automated discovery. *arXiv preprint arXiv:2602.23413*, 2026.

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.

Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune. Towards end-to-end automation of ai research. *Nature*, 651(8107):914–919, 2026.

Robert Lupoiu, Yixuan Shao, Tianxiang Dai, Chenkai Mao, Kofi Edée, and Jonathan A Fan. A multi-agentic framework for real-time, autonomous freeform metasurface design. *Science Advances*, 11 (44):eadx8006, 2025.

Jiankun Lyu, Nicholas Kapolka, Ryan Gumpper, Assaf Alon, Liang Wang, Manish K Jain, Ximena Barros-Álvarez, Kensuke Sakamoto, Yoojoong Kim, Jeffrey DiBerto, et al. Alphafold2 structures guide prospective ligand discovery. *Science*, 384(6702):eadn6354, 2024.

Jun Ma, Feifei Li, and Bo Wang. U-mamba: Enhancing long-range dependency for biomedical image segmentation. *arXiv preprint arXiv:2401.04722*, 2024.

Indrajeet Mandal, Jitendra Soni, Mohd Zaki, Morten M Smedskjaer, Katrin Wondraczek, Lothar Wondraczek, Nitya Nand Gosvami, and NM Anoop Krishnan. Evaluating large language model agents for automation of atomic force microscopy. *Nature Communications*, 16(1):9104, 2025.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Deepak Nathani, Lovish Madaan, Nicholas Roberts, Nikolay Bashlykov, Ajay Menon, Vincent Moens, Amar Budhiraja, Despoina Magka, Vladislav Vorotilov, Gaurav Chaurasia, et al. MLGym: A new framework and benchmark for advancing ai research agents. *arXiv preprint arXiv:2502.14499*, 2025.

Eric Nguyen, Michael Poli, Matthew G Durrant, Brian Kang, Dhruva Katrekar, David B Li, Liam J Bartie, Armin W Thomas, Samuel H King, Garyk Brixi, et al. Sequence modeling and design from molecular to genome scale with evo. *Science*, 386(6723):eado9336, 2024.

R OpenAI. Gpt-4 technical report. arxiv 2303.08774. *View in Article*, 2(5), 2023.

Maciej P Polak and Dane Morgan. Extracting accurate materials data from research papers with conversational language models and prompt engineering. *Nature Communications*, 15(1):1569, 2024.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Saikat Roy, Gregor Koehler, Constantin Ulrich, Michael Baumgartner, Jens Petersen, Fabian Isensee, Paul F Jaeger, and Klaus H Maier-Hein. Mednext: transformer-driven scaling of convnets for medical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 405–415. Springer, 2023.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.

Sandra Steyaert, Marija Pizurica, Divya Nagaraj, Priya Khandelwal, Tina Hernandez-Boussard, Andrew J Gentles, and Olivier Gevaert. Multimodal data fusion for cancer biomarker discovery with deep learning. *Nature machine intelligence*, 5(4):351–362, 2023.

Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014.

Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. Monte carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review*, 56(3): 2497–2562, 2023.

Yucheng Tang, Dong Yang, Wenqi Li, Holger R Roth, Bennett Landman, Daguang Xu, Vishwesh Nath, and Ali Hatamizadeh. Self-supervised pre-training of swin transformers for 3d medical image analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 20730–20740, 2022.

NovelSeek Team, Bo Zhang, Shiyang Feng, Xiangchao Yan, Jiakang Yuan, Zhiyin Yu, Xiaohan He, Songtao Huang, Shaowei Hou, Zheng Nie, et al. Novelseek: When agent becomes the scientist–building closed-loop system from hypothesis to verification. *arXiv preprint arXiv:2505.16938*, 2025.

Edan Toledo, Karen Hambardzumyan, Martin Josifoski, Rishi Hazra, Nicolas Baldwin, Alexis Audran-Reiss, Michael Kuchnik, Despoina Magka, Minqi Jiang, Alisia Maria Lupidi, et al. Ai research agents for machine learning: Search, exploration, and generalization in mle-bench. *arXiv preprint arXiv:2507.02554*, 2025.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Frank Verstraete, Valentin Murg, and J Ignacio Cirac. Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems. *Advances in Physics*, 57(2):143–224, 2008.

Tianze Wang, Bowen R Qin, Sihong Li, Zimo Wang, Xuejian Li, Yuanxu Jiang, Chenrui Qin, Qi Ouyang, Chunbo Lou, and Long Qian. Discovery of diverse and high-quality mrna capping enzymes through a language model–enabled platform. *Science advances*, 11(15):eadt0402, 2025.

David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

Zhong-Cheng Wu, Ting-Zhu Huang, Liang-Jian Deng, Hong-Xia Dou, and Deyu Meng. Tensor wheel decomposition and its tensor completion application. In *Advances in Neural Information Processing Systems*, 2022.

Hongliang Xin, John R Kitchin, and Heather J Kulik. Towards agentic science for advancing scientific discovery. *Nature Machine Intelligence*, pp. 1–3, 2025.

Ilya Yakavets, Sina Kheiri, Jennifer Cruickshank, Riley J Hickman, Faeze Rakhshani, Matteo Aldeghi, Ella M Rajaonson, Edmond WK Young, Alán Aspuru-Guzik, David W Cescon, et al. Machine learning-assisted exploration of multidrug-drug administration regimens for organoid arrays. *Science Advances*, 11(31):eadt1851, 2025.

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.

Liang Yan, Shengzhong Zhang, Bisheng Li, Min Zhou, and Zengfeng Huang. Unreal: Unlabeled nodes retrieval and labeling for heavily-imbalanced node classification. *arXiv preprint arXiv:2303.10371*, 2023.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023.

Xu Yang, Xiao Yang, Shikai Fang, Bowen Xian, Yuante Li, Jian Wang, Minrui Xu, Haoran Pan, Xinpeng Hong, Weiqing Liu, Yelong Shen, Weizhu Chen, and Jiang Bian. R&D-Agent: Automating data-driven AI solution building through LLM-powered automated research, development, and evolution. *arXiv preprint arXiv:2505.14738*, 2025.

Tianxiang Zhao, Xiang Zhang, and Suhang Wang. Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *Proceedings of the 14th ACM international conference on web search and data mining*, pp. 833–841, 2021.

Lucas Zimmer, Marius Lindauer, and Frank Hutter. Auto-pytorch: Multi-fidelity metalearning for efficient and robust autodl. *IEEE transactions on pattern analysis and machine intelligence*, 43(9): 3079–3090, 2021.

A Prompts of SelfAI System

SelfAI separates task formalization, trajectory analysis and hypothesis generation, strategic planning, stopping judgment, and experiment execution. The Experiment Manager executes selected experiments through existing training and evaluation tools and returns their results. In the controlled benchmark, each configuration has already been executed, so it returns the recorded execution result and all methods receive identical outcomes. The framework does not retrain its foundation-model backbone or autonomously expand the user-authorized candidate space.

A.1 Metric computation and aggregation

The AUP_D implementation uses positive finite objectives and the distinct log values in $\mathcal{R}_i$ as shared thresholds. All integrals use the composite trapezoidal rule. Endpoint equality is included and repeated trajectory values remain separate trials. Because the calculation depends on the multiset of observed values rather than their order, permuting an otherwise identical trajectory leaves AUP_D unchanged. Discovery and stopping order are instead evaluated by t_{best}, t_{stop}, and Score. A degenerate reference range gives $A_i = 0$ and $S_i' = 1$. Empty trajectories, invalid values, undefined baseline moments, and nonfinite results are marked missing. Normalization uses the designated baseline on the same tasks and references. GS is the baseline in the controlled benchmark.

The comparison in the main results excludes LCBench. Its complete table is $\mathcal{R}_i$ and its size is H_i. A run missing the reference best receives $t_{\mathrm{best}} = 1$. Randomized methods are run with multiple random seeds. Metrics are averaged first across seeds within problem and then equally across problems, producing Table 3. After all methods have run, $\mathcal{R}_i$ is updated by combining their recorded search spaces. This union gives every method the same reference within a comparison.

A.2 Prompt of User Agent: Idea Interaction and Experiment Configuration

User Agent Prompt for Configuration Interaction

System: ...
User:
{{SUMMARIZED_CONTENT}}
Please fill out the content following the YAML format:

```yaml
# Template for Configuration Interaction
# to convert Idea Interaction into YAML format
- role: system
  content:
    model: $modelName
    description: You are a $role specializing in
                 studying $taskName. Please provide
                 professional and detailed answers.
    task: $taskName
    basic_idea: $basic_idea
    search_space:
      $hyper_name1: []
      $hyper_name2: []
      ...
    link: $paperLink
    instruction: Complete instructions under limited trials.
- role: user
  content:
    max_trials: $maxTrials
    trials: []
```

A.3 Reasoning Prompt for Iterative Experimentation

A.3.1 Trajectory Analysis and Hypothesis Generation

The Cognitive Agent begins with **Task 1: Analyze the Current Task**, which identifies the objective, constraints, and experimental parameters. In the controlled benchmark, these parameters specify the admissible configurations.

In **Task 2: Analysis of Completed Trials**, the agent summarizes observed parameter–performance associations and identifies directions for further evaluation. This step performs hypothesis generation: it generates the candidate experiments to be selected, namely configurations expected to perform well or to be informative given the observed trends. Strategic planning (Sect. A.3.3) then selects among them, and the benchmark evaluates the resulting choices and stopping decisions.

Cognitive Prompt for Trajectory Analysis and Hypothesis Generation

System:
{{SYSTEM_PART_OF_CONFIGURATION}}
User: Completed trials:
{{COMPLETED_TRIALS}}
Task 1: Analyze the current task
Understand current tasks, basic ideas, objectives, and hyperparameters.
Task 2: Analysis of Completed Trials
Step 1: Summarize performance metrics for completed trials.
Step 2: Evaluate performance trends for hyperparameters.
Step 3: Highlight promising hyperparameter combinations.

A.3.2 STOPPING JUDGMENT

We introduce a stopping judgment to guide prompt designs that balance exploration and exploitation. During the stopping judgment, the Cognitive Agent first determines whether the current performance metrics surpass those of the initial configuration and then judges whether to stop. Building on the prior analysis of completed trials, the agent systematically evaluates stopping conditions to avoid testing all configurations.

By using prompt instructions, the stopping judgment evaluates all completed trials by leveraging insights from trial analysis, observed performance trends, and identified key findings. In practice, we implement this judgment process via a structured prompt template comprising two main tasks and explicit stopping criteria. The language model then analyzes the relationship between completed trials and unexplored areas. The prompt is organized as follows:

Cognitive Prompt for Best Stopping Judgment

User: Completed trials:
`{{COMPLETED_TRIALS}}`
The following **Search Space** contains **unexplored** trials. `{{TRIALS}}`
Instructions: Task 1: Review Analysis of Completed Trials (trial analysis, performance trends, highlights, and other insights)
Task 2: Decide Whether to Stop Optimization
Based on the above analysis and **Completed Trials**, determine whether the optimization process should be stopped.
Carefully analyze each of the following stop rules and provide a short (1-2 sentences) justification for whether it is met:
1. Have all promising configurations identified based on performance trends been tested?
2. Is it unlikely that unexplored configurations will perform better based on the observed trends and the law of diminishing returns?
3. Has the best metric improved significantly?
Step 2: Decide whether **all** conditions are met.
If **all** criteria in Step 1 are met, Answer: Yes, with confidence score: `{{CONFIDENCE_SCORE}}`. Otherwise, Answer: No with confidence score: `{{CONFIDENCE_SCORE}}`.

A.3.3 STRATEGIC PLANNING

This prompt implements the strategic planning stage of Cognitive Agent. Its objective is to generate candidate experimental configurations for the next iteration based on accumulated experimental evidence, while balancing refinement of high-performing configurations and exploration of under-sampled regions.

Cognitive Prompt for Strategic Planning

User: Instructions:
Task 1: Review Analysis of All Completed Trials
Completed trials:
`{{COMPLETED_TRIALS}}`
The following **Search Space** contains **unexplored** trials:
`{{TRIALS}}`
Instructions:
Task 2: Optimization Recommendation
Recommend exactly `{{N_JOBS}}` promising trials from the provided **Search Space** (include both number and params).
Rules:
1. "params" MUST include:
`{{HYPERNAME}}`

Table 5: Search-policy comparison under the same stopping choices. #Trials is the average number of executed trials.

Method	Search policy	Stopping	Score↑	AUP_D ↓	t_{best} ↓	t_{stop} ↓	Best Norm↑	Hit Rate↑	#Trials↓
SelfAI	heuristic + LLM	SelfAI judgment	**0.6019**	0.2647	**0.3642**	0.4313	0.9981	0.9231	12.92
LLM-MCTS + SelfAI stop	LLM expansion + UCT	SelfAI judgment	0.3554	0.4448	0.6673	0.5816	0.9513	0.5385	17.54
LLM-MCTS-ES	LLM expansion + UCT	patience	0.3571	**0.1724**	0.7867	**0.3411**	0.8557	0.3077	**9.38**
MCTS-ES	heuristic + UCT	patience	0.4401	0.2322	0.6137	0.4043	0.9016	0.5385	10.31

2. All selected ‘params‘ must match exactly with the provided **Search Space**. Do NOT leave out any key.
3. Use the analysis in **Task 1** (trial analysis, performance trends, highlights, and other insights) to guide selection.
4. Based on the above analysis, explore under-explored regions only when there is clear evidence of potential performance gain.
5. Do not mix, modify, or create new values.
6. You MUST not output any JSON blocks in this part.
7. You MUST provide reasoning for each recommendation.

B Details of SelfAI Benchmark

In this section, we summarize the averaged performance across all benchmark tasks. Classical baselines, including grid search and TPE-based Bayesian optimization, achieve competitive final objective values but obtain low Scores and high AUP_D, reflecting redundant late-stage exploration and delayed stopping (Table 3). Naive LLM solvers improve early discovery in some tasks but exhibit substantial variability across domains. In contrast, SelfAI-driven solvers based on mid-sized models, such as Qwen2.5-7b and GPT4-o3-mini, consistently achieve the highest Scores with substantially lower AUP_D and earlier stopping times, indicating more focused and efficient discovery trajectories. Larger models (e.g., Qwen2.5-72b and Llama3.3-70b) do not necessarily yield improved efficiency, highlighting that trajectory-aware reasoning within SelfAI plays a more critical role than model scale. For completeness, we discuss representative atypical trajectory behaviors in Supplementary Sects. C and D, where corresponding examples are provided. These cases illustrate exploration behaviors and breakdown patterns observed in SelfAI Benchmark, including premature stopping, limited integration of earlier observations, and sensitivity to minor perturbations, and serve to contextualize the aggregate performance trends reported above.

B.1 Baselines

All methods share the same candidate space, initial trials, observation-history format, unique-trial budget, and YAML output schema. Classical baselines are Grid Search (GS), TPE Bayesian Search (BS) (Akiba et al., 2019), SMAC3 (Lindauer et al., 2022), Random Search (Bergstra & Bengio, 2012), Successive Halving (SHA) (Jamieson & Talwalkar, 2016), and MCTS (Świechowski et al., 2023), which expands the search tree heuristically and selects trials by UCT. The LLM-based baselines are defined as follows. **LLM** directly generates experimental configurations from the task description without iterative feedback. **LLM-ES** receives the same completed and unexplored trials and the same stop-or-continue action space as SelfAI, but not SelfAI's trajectory-analysis and strategic-planning context. **OPRO** (Yang et al., 2023) prompts the LLM with previously evaluated configurations and their scores to propose the next configuration. **LLM-MCTS** uses the LLM to expand the MCTS tree and UCT to select trials. Stopping-enabled baselines make an online stop-or-continue decision from completed-trial history: Random-ES, BS-ES, MCTS-ES, and LLM-MCTS-ES stop by patience; MCTS-UCT-stop stops by a UCT criterion; LLM-ES and LLM-MCTS + SelfAI stop use the SelfAI stopping judgment. Rule-SelfAI keeps SelfAI's trajectory analysis and strategic planning but replaces the SelfAI stopping judgment with patience stopping (SelfAI w/ patience stopping in Table 4), and SelfAI w/o stopping runs SelfAI to the full budget. All LLM-based methods use the GPT4-o3-mini backbone in the main comparison. Supplementary Table 5 compares search policies under the same stopping choices.

Table 6: Controlled benchmark tasks. "No. Configs" is the number of available candidate configurations with completed outcome records, not the number selected by every solver.

Category	Method	Task	Dim	No. Configs	Ref.
Scientific Computing	Tensor Network	Image Completion	3	64	Wu et al. (2022)
Machine Learning	Random Forest	Pricing Prediction	5	162	Huang (2020)
	LSTM	Sentiment Analysis	2	20	Kandhro et al. (2020)
	GraphSAGE	Node Classification	22	25	Yan et al. (2023)
Computer Vision	SIREN	Image Denoising	2	25	Sitzmann et al. (2020)
	SIREN	Image Segmentation	2	25	Sitzmann et al. (2020)
	ResNet	Image Classification	4	9	He et al. (2016)
	MAE	Image Classification	2	20	He et al. (2022)
	LCBench-NN	Image Classification	5	2000	Zimmer et al. (2021)
Medical Image Analysis	nnUnet	BraTS LaBella et al. (2023)	3	18	Isensee et al. (2021)
	nnUnet-revisited	BTCV Landman et al. (2015)	5	19	Isensee et al. (2024)
Drug Discovery	DNN	Bioactivity Prediction	4	30	Korotcov et al. (2017); Ekins et al. (2015)

B.2 Machine Learning

Boston house pricing prediction. We evaluate SelfAI on the Boston housing price prediction task Huang (2020) using a random forest regression model. The candidate space contains 162 configurations defined by five hyperparameters: i.e., n-estimators $= [100, 200, 300]$, max-depth $=$ [None, 10, 20], min-samples-split $= [2, 5, 10]$, min-samples-leaf=[1,2,5] and max-features $=$ ["sqrt", "log2"]).

As shown in Supplementary Fig. 4 and Supplementary Table 9, GPT4-o3-mini achieves the highest overall Score (0.9811), ranking first among all evaluated solvers. This performance reflects its ability to rapidly identify high-performing configurations while terminating exploration shortly thereafter, yielding superior optimization efficiency. Within the DeepSeek-R1 family, the 32B and 7B variants rank second and third, respectively, exhibiting a favorable balance between early convergence (low t_{best}) and moderate performance diversity (AUP_D) among open-source models. In contrast, larger models such as DeepSeek-R1-70b and Llama3.3-70b stop prematurely, terminating the search before reaching the best configuration. Notably, nearly all solvers reach an identical and high Best Result (approximately 0.841), indicating that differences in final predictive accuracy are minimal. Instead, the primary distinction lies in how efficiently and reliably solutions are discovered, underscoring the importance of trajectory-aware reasoning and adaptive stopping in practical scientific optimization.

Sentiment analysis. We perform experiments for sentiment analysis that focus on identifying opinions, emotions, and attitudes expressed in text. All experiments are found in Kandhro et al. (2020) with standard experimental settings, where pre-trained Word2Vec embeddings Mikolov et al. (2013) are used as input features. An LSTM network is then employed to model sentence sequences, converting them into a multi-class classification problem. This setup serves to evaluate the agent's reasoning and optimization capabilities in textual domains. As reported in Supplementary Table 10, GPT4-o3-mini achieves the highest Score (0.8824), consistently identifying the optimal configuration at an early stage while maintaining a balanced level of exploration. In contrast, the DeepSeek-R1 and Qwen2.5 series exhibit substantially lower Scores, frequently showing delayed convergence or failing to reach the global optimum. Notably, GPT4-o3 performs markedly worse (Score 0.2745, rank 9). This result indicates that model size alone does not guarantee effective iterative hyperparameter reasoning in recurrent neural architectures, and that disciplined trajectory reasoning combined with adaptive stopping plays a more decisive role in efficient optimization within textual learning tasks.

B.2.1 Scientific Computing

We selected the tensor decomposition method Wu et al. (2022). The method is an important tool for high-dimensional data analysis and is crucial in applications such as data compression Li et al. (2025), computational acceleration Berezutskii et al. (2025), and multi-modal data fusion Steyaert et al. (2023). All LLMs are provided with identical mathematical knowledge about TW decomposition generated by GPT-4o. As shown in Supplementary Table 11, Qwen2.5 (7b and 14b) and GPT4-o3-mini perform well and rank highly. In contrast, Qwen2.5-72b exhibits reduced performance, suggesting that broader general reasoning alone may be insufficient to navigate the specific trade-offs inherent in tensor decomposition. In the DeepSeek series, although the DeepSeek-R1-7b model can't find the optimal solution, its search strategy was acceptable with a moderate Score, demonstrating its

ability to perform mathematical reasoning. DeepSeek-R1-70b and GPT4-o3, despite excellent search diversity and early stopping, received a mediocre score, reflecting that their exploration strategy was not well aligned with the optimization goal of tensor decomposition, possibly failing to find the optimal performance between data compression and computational efficiency.

B.3 Computer Vision

A. SIREN We use SIREN (Sinusoidal Representation Networks) (Sitzmann et al., 2020) for image segmentation and denoising. Its performance varies with learning rate and regularization strength, providing a controlled setting for examining how observed parameter–performance associations guide subsequent trials and stopping.

Figure 2c illustrates hyperparameter search trajectories for image segmentation using SIREN (see also Supplementary Figs. 5, 6, and 7), where the surface, smoothed from the original steep, multi-peak data, reveals distinct search behaviors. Given the same three initial points, the tree-structured Bayesian optimizer (BS) follows a spiral trajectory that underutilizes promising starting regions and fails to explore broadly, reaching the best configuration only later. In contrast, SelfAI infers trends from evaluated points and explores more broadly to locate the global optimum efficiently. It also monitors progress and halts when improvements plateau.

Quantitative results (Supplementary Tables 12-13) reveal distinct task-dependent performance profiles. In segmentation, DeepSeek-R1-7b achieves the highest Score (0.693) and ranks first, followed by Qwen2.5-7b and GPT4-o3-mini. Larger models such as Qwen2.5-72b and Llama3.3-70b rank only mid-range, indicating that scale alone does not ensure effective optimization. In denoising, the landscape shifts: Qwen2.5-72b delivers the best performance (Score 0.761), while DeepSeek-R1-14b and DeepSeek-R1-32b jointly occupy second place. Notably, models that perform well in segmentation tasks do not necessarily perform well in denoising tasks, suggesting that the effectiveness of the solver largely depends on the degree of matching between its inference strategy and the task.

B. Image Classification We also evaluate Masked Autoencoder (MAE) He et al. (2022) and ResNet He et al. (2016). Their configuration spaces provide additional settings for comparing trial allocation and stopping under the same observation protocol.

Masked Autoencoder (MAE) MAE introduces two key hyperparameters: training strategies (linear detection vs. fine-tuning) and masking rate, which are explored over a range: $[0.10, 0.90]$ with an interval of 0.1. The masking rate directly affects the difficulty of the reconstruction task and the quality of the learning representation, while the training strategy determines how to adapt to the downstream task. In our experiments (see Supplementary Table 14 for more details), GPT4-o3-mini achieved explicit performance with efficiency, demonstrating the extraordinary ability to identify the best mask configuration and training strategies. Qwen2.5-7b achieves early convergence and disciplined stopping. The DeepSeek-R1 series shows consistent scaling behavior: performance improves from 7B to 14B to 32B, but slightly drops at 70B. In traditional solvers, GS and BS lag significantly behind LLM-based solvers, highlighting the advantages of LLM-enabled discovery. Apart from Qwen2.5-14b and DeepSeek-R1-7b, most solvers can find the optimal solution, but the methods differ substantially in search efficiency and convergence speed.

ResNet Family On the ImageNet ResNet hyperparameter benchmark, LLMs in the 7B-32B range demonstrate superior optimization efficacy. The top-performing solvers, such as Qwen2.5-14b (1st), DeepSeek-R1-32b (2nd), and Qwen2.5-7b with DeepSeek-R1-14b (tied 3rd), consistently identify the optimal ResNet configuration within one or two trials, halting immediately with near-perfect sample efficiency. In contrast, larger variants from the same model families drop to middle or lower rankings, consuming half or more of the search budget with notably higher exploration overhead. While GPT-series models, as large-scale counterparts, still achieve competitive results, the 7B-32B class exhibits a clear advantage in balancing accuracy and efficiency. For the architectural search on ImageNet, we explored key dimensions such as depth and bottleneck design. Qwen2.5-14b attains theoretical peak performance, with its 7B and 32B versions completing further exploration efficiently. These results reflect an ability to identify ideal network configurations with minimal wasted exploration.

Larger candidate-space benchmark The supplementary LCBench evaluation Zimmer et al. (2021) uses a table of 2000 configurations covering learning rate, batch size, network depth, and dropout

rate (Supplementary Fig. 3). This is the size of the available candidate set; each solver selects its own subset. GPT4-o3-mini and GPT4-o3 were omitted because the task representation exceeded the context limits used in this evaluation. The plots show interactions among parameters and localized high-performing regions. Supplementary Table 16 compares endpoint quality, time to the reference best, and stopping across backbones. This larger-space study is reported separately from the aggregate in Table 3.

B.4 Medical Image Analysis

In medical image analysis, nnU-Net Isensee et al. (2021) is a landmark framework known for its strong generalization and automated design. Despite its adaptive network structure, preprocessing, and training strategies for various segmentation tasks, the exploration of novel architectures persists. To address this, nnU-Net-Revisited Isensee et al. (2024) establishes a comprehensive benchmark including 19 mainstream models (CNN-based Ronneberger et al. (2015); Isensee et al. (2021); Roy et al. (2023), Transformer-based Vaswani et al. (2017); Tang et al. (2022), and Mamba-based Gu & Dao (2023); Ma et al. (2024)), offering a solid basis for fair evaluation. We evaluate SelfAI on this benchmark across multiple datasets, with results summarized in Supplementary Fig. 4 and Supplementary Tables 17-18. On the BTCV dataset Landman et al. (2015), SelfAI instantiated with GPT4-o3-mini achieves the highest Score (0.6875) and ranks first, indicating rapid identification of high-quality configurations combined with efficient stopping behavior. Notably, Qwen2.5-7b secures second place, outperforming all larger variants within the same model family, suggesting that effective optimization in this setting does not scale monotonically with model size. A similar pattern emerges in nnU-Net hyperparameter tuning on the BraTS dataset LaBella et al. (2023). Qwen2.5-14b attains the top ranking (Score 0.4333), followed closely by GPT4-o3-mini, while Qwen2.5-32b and DeepSeek-R1-70b tie for third. On both datasets, performance does not scale monotonically with model size, reinforcing the observation that increased model capacity alone does not guarantee superior optimization performance. These results highlight the importance of trajectory-aware decision-making under noisy and high-variance evaluation regimes, which are characteristic of medical image analysis. By committing early to promising configurations while avoiding prolonged exploration with diminishing returns, SelfAI enables efficient discovery in settings where excessive experimentation is both computationally costly and practically constrained.

B.5 Imbalanced Node Classification

Graph neural networks (GNNs) are a central modeling paradigm for relational data and have become indispensable across a range of scientific domains. In biomedicine, GNNs underpin modern protein structure prediction systems, such as AlphaFold3, by enabling explicit modeling of spatial interactions among amino acid residues, thereby advancing structural biology at unprecedented resolution Abramson et al. (2024). In drug discovery, GNNs are widely applied to molecular property prediction, compound-target interaction modeling, and de novo molecule generation Bongini et al. (2021). Despite their success, optimizing GNNs on imbalanced datasets remains particularly challenging, due to sparse and asymmetric reward signals, complex relational dependencies, and high-dimensional hyperparameter spaces. These characteristics make such tasks especially sensitive to exploration strategy and stopping decisions. To investigate whether trajectory-aware reasoning can address these challenges, we evaluate GraphSAGE on the imbalanced Cora benchmark Zhao et al. (2021) under the setup Yan et al. (2023).

As shown in Supplementary Fig. 4, SelfAI instantiated with the GPT4-o3 series consistently achieves near-optimal performance, characterized by rapid identification of optimal configurations, efficient stopping decisions, and sustained exploration diversity. This behavior indicates that SelfAI effectively balances exploitation and exploration in structurally complex optimization landscapes. In contrast, Qwen2.5 models (with the exception of the 72b variant) exhibit stable but middling performance across scales, while larger models tend to converge later and incur substantial redundant exploration. This pattern suggests potential limitations in capturing the structural optimization requirements of relational domains such as molecular graphs and protein interaction networks (see Supplementary Table 19). These results provide evidence that trajectory-level reasoning and adaptive stopping generalize beyond continuous parameter optimization, extending effectively to structured, relational learning problems that are common in real-world scientific applications.

B.6 Drug Discovery

Drug discovery increasingly benefits from advances in machine learning and data-driven modeling, which serve as versatile frameworks for tasks such as bioactivity prediction, virtual screening, and compound prioritization. These approaches have demonstrated the potential to accelerate early-stage discovery, reduce experimental burden, and shorten development timelines Brahmavar et al. (2024); Grisoni et al. (2021). Following the evaluation practices and modeling standards summarized by Korotcov et al. Korotcov et al. (2017), this study employs the Chagas EP20 dataset Ekins et al. (2015), which measures the activity of compounds against Trypanosoma cruzi, a neglected but medically significant target in tropical disease research. The core challenge lies in learning an effective mapping from molecular representations, such as SMILES sequences Weininger (1988), to experimentally measured bioactivity Fliri et al. (2005). We adopt SelfAI to adaptively refine model performance, monitor discovery trajectories, and determine principled stopping criteria for iterative optimization. Experimental results (see Supplementary Table 20) reveal distinct performance patterns among solvers in hyperparameter optimization. SelfAI and select LLM-based solvers achieve superior Scores by rapidly identifying promising hyperparameter regions and terminating exploration once gains plateau. Notably, these results demonstrate efficient optimization under realistic biochemical uncertainty, highlighting trajectory-aware optimization as a practical computational enabler for scalable and resource-efficient bioactivity modeling in drug discovery pipelines.

B.7 Open-ended evaluation on MLE-Bench

MLE-Bench (Chan et al., 2024) combines candidate generation, code implementation, debugging, execution, repeated selection, and stopping. Unlike the controlled benchmark, its runnable candidate set evolves over time. At step t, the native agent proposes code artifacts and execution outcomes define the currently available set $\mathcal{X}_t$. We retain the native generation/debug/execution stack of AIDE (Jiang et al., 2025) and ML-Master, and use SelfAI only to choose among draft/debug/improve actions and decide whether to continue. This design does not isolate SelfAI and is not used as an attribution test. Instead, it shows why endpoint outcomes in open workflows can misrepresent exploration: the quality and correctness of the candidate code, the reliability of its execution, and the fidelity of validation can dominate the final score and also alter later decisions. Plant and Jigsaw use AUC (higher is better), while Taxi uses RMSE (lower is better). Validation (Val) guides search. Private is the MLE-Bench grade of the Val-best submission and is used only to assess generalization. R&D-Agent (Yang et al., 2025) under its official configuration did not finish the three runs reliably and is not reported.

Table 7: Native versus SelfAI-augmented open-ended agents on MLE-Bench (20 steps). Higher is better for AUC and lower is better for RMSE.

Task	System	Native Val	Native Private	SelfAI Val	SelfAI Private
Plant AUC↑	AIDE	**0.9805**	0.9594	0.9794	**0.9677**
Plant AUC↑	ML-Master	0.9783	**0.9964**	**0.9799**	0.9943
Taxi RMSE↓	AIDE	4.6124	4.5084	**4.4557**	**4.4305**
Taxi RMSE↓	ML-Master	**3.2823**	**4.8076**	3.4381	6.5371
Jigsaw AUC↑	AIDE	**0.9942**	0.9573	0.9876	**0.9781**
Jigsaw AUC↑	ML-Master	**0.9877**	**0.9800**	0.9749	0.9628

Wins are split. SelfAI improves validation in two of six system–task pairs (ML-Master Plant and AIDE Taxi) and the private score in three (AIDE on all three tasks), but ML-Master becomes worse on Taxi private RMSE (4.81 to 6.54) and on both Jigsaw scores. Because SelfAI does not write or repair code, its selection and stopping decisions depend on the candidates and results produced by the native agent. Conversely, an execution failure or an unreliable validation result changes the history on which later SelfAI decisions are based. The mixed outcomes therefore reflect the host agent's generation, execution, and validation as much as SelfAI's decisions, and neither the gains nor the losses can be attributed to SelfAI from the final score alone.

Table 8: Longer AIDE runs (100 steps or 24 hours). "Valid/attempted" counts usable validation scores. Plant native reached the time limit after 70 attempts. The other runs completed 100.

Task	Variant	Valid/attempted	Val	Private
Plant AUC↑	native	57/70	0.9955	**0.9969**
Plant AUC↑	SelfAI	68/100	**0.9959**	0.9814
Taxi RMSE↓	native	81/100	5.1465	914.66
Taxi RMSE↓	SelfAI	74/100	**3.5633**	**31.99**

The longer runs expose validation–test mismatch: SelfAI is slightly better on Plant Val but worse on Private, while Taxi Val improves despite severe Private failure. Because validation guides later choices and stopping, such errors also alter the subsequent trajectory. Across both run lengths, the endpoint therefore depends on three factors besides SelfAI's decisions: the quality and correctness of the candidate code, the reliability of its execution, and the fidelity of validation. Endpoint scores thus conflate good exploration with good code. The controlled benchmark fixes these three factors, so that only selection and stopping differ between methods, which allows exploration and discovery to be evaluated accurately. These open-ended results are therefore reported separately from Tables 3–4.

C Case studies

The following case studies illustrate how solvers use accumulated observations, allocate trials, and stop in the controlled benchmark. They connect the aggregate trajectory metrics to individual runs and show cases in which similar endpoint quality accompanies different experimental costs. Task-specific analyses are provided in Supplementary Sect. B.

Machine learning. We first evaluate SelfAI on the Boston housing price prediction task Huang (2020) using a random forest regression model. As shown in Supplementary Fig. 4 and Supplementary Table 9, nearly all solvers achieve a similar and high best result (≈ 0.841), indicating that differences in final predictive accuracy are minimal. Small- and medium-sized models outperformed other solvers, whereas the large models DeepSeek-R1-70b and Llama3.3-70b stopped prematurely and missed the best configuration. These results illustrate the importance of trajectory-aware reasoning and adaptive stopping in practical scientific optimization, as larger-scale models may exhibit either overly diffuse exploration or premature stopping and are not always aligned with task-specific objectives. Further examples are presented below in this section.

Computer vision. Fig. 2c illustrates these trajectory-level dynamics in the hyperparameter search landscape for image segmentation using SIREN (see also Supplementary Figs. 5-7). The smoothed response surface reveals several obvious peaks, indicating multiple competing high-performance regions. Starting from identical initial points, the tree-based Bayesian optimizer follows a spiral-like trajectory that underutilizes promising regions and exhibits limited global coverage, reaching the best configuration only after many trials. In contrast, SelfAI infers trends from accumulated evaluations and adapts exploration accordingly, resulting in broader coverage of the search space and earlier identification of globally competitive configurations. Importantly, SelfAI also monitors trajectory-level progress and terminates exploration once improvements plateau. This behavior directly translates into higher scores and lower AUP_D, demonstrating how trajectory-aware metrics capture both discovery efficiency and controlled diversity beyond final performance alone.

Medical image analysis. The nnU-Net Isensee et al. (2021) and nnU-Net-Revisited Isensee et al. (2024) tasks compare predefined configurations on BraTS LaBella et al. (2023) and BTCV Landman et al. (2015). These small candidate spaces allow inspection of individual selection and stopping decisions. Supplementary Fig. 4 and Tables 17–18 show that similar final segmentation performance can accompany different times to the best reference configuration and different trial counts. The trajectory metrics reveal these differences, including prolonged exploration after the best result has been reached.

Scientific computing. Tensor decomposition Hackbusch (2014); Verstraete et al. (2008) provides a further configuration-selection task Wu et al. (2022). Supplementary Table 11 shows that both prolonged exploration and premature stopping can reduce Score. A solver may sample diverse performance values yet miss the best reference configuration, or reach a useful result early and con-

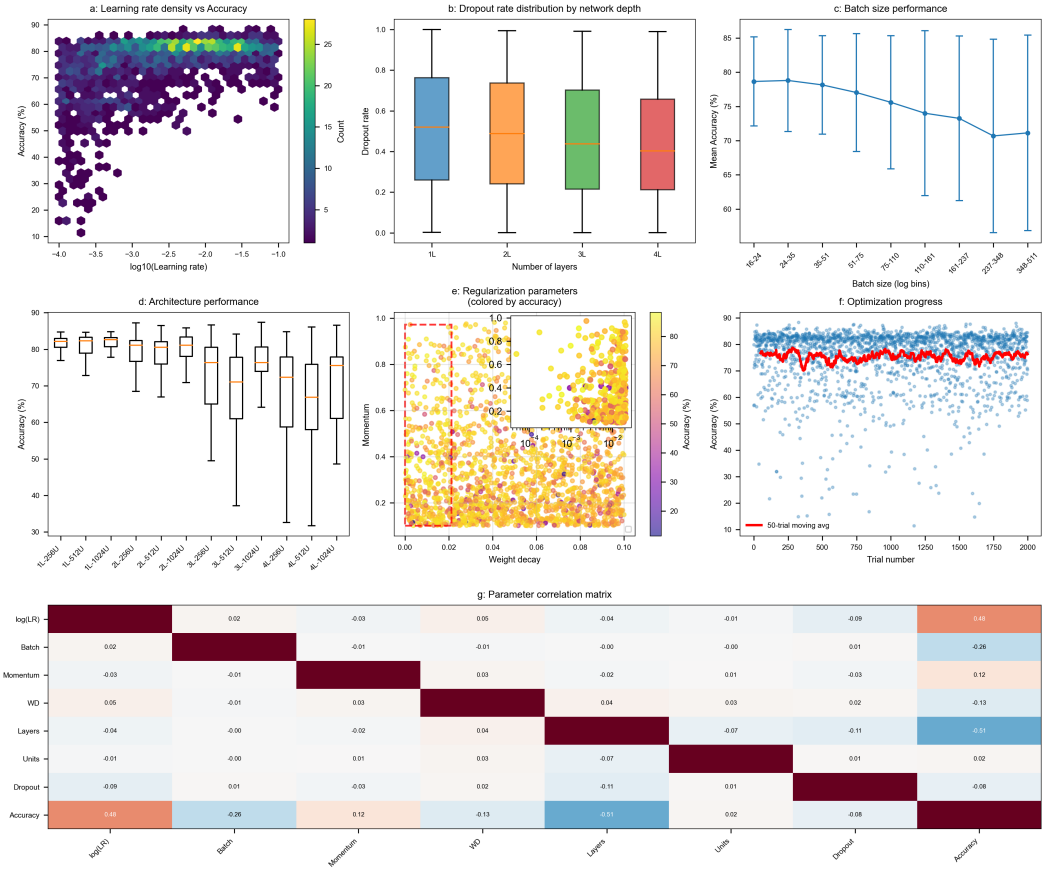

Figure 3: The relationship between different parameter settings and accuracy. **a**, A hexbin plot showing the joint density of learning rate (log-scaled) and accuracy. **b**, Box plots illustrating the distribution of dropout rates used for models of different depths. **c**, Mean accuracy and standard error across logarithmically binned batch sizes. **d**, Performance comparison of different architectural configurations (layers and units). **e**, A scatter plot of weight decay against momentum, colored by accuracy. **f**, The optimization trajectory, showing accuracy improvement over successive trials, with a moving average trend line. **g** A correlation matrix quantifying linear relationships between all parameters and the accuracy.

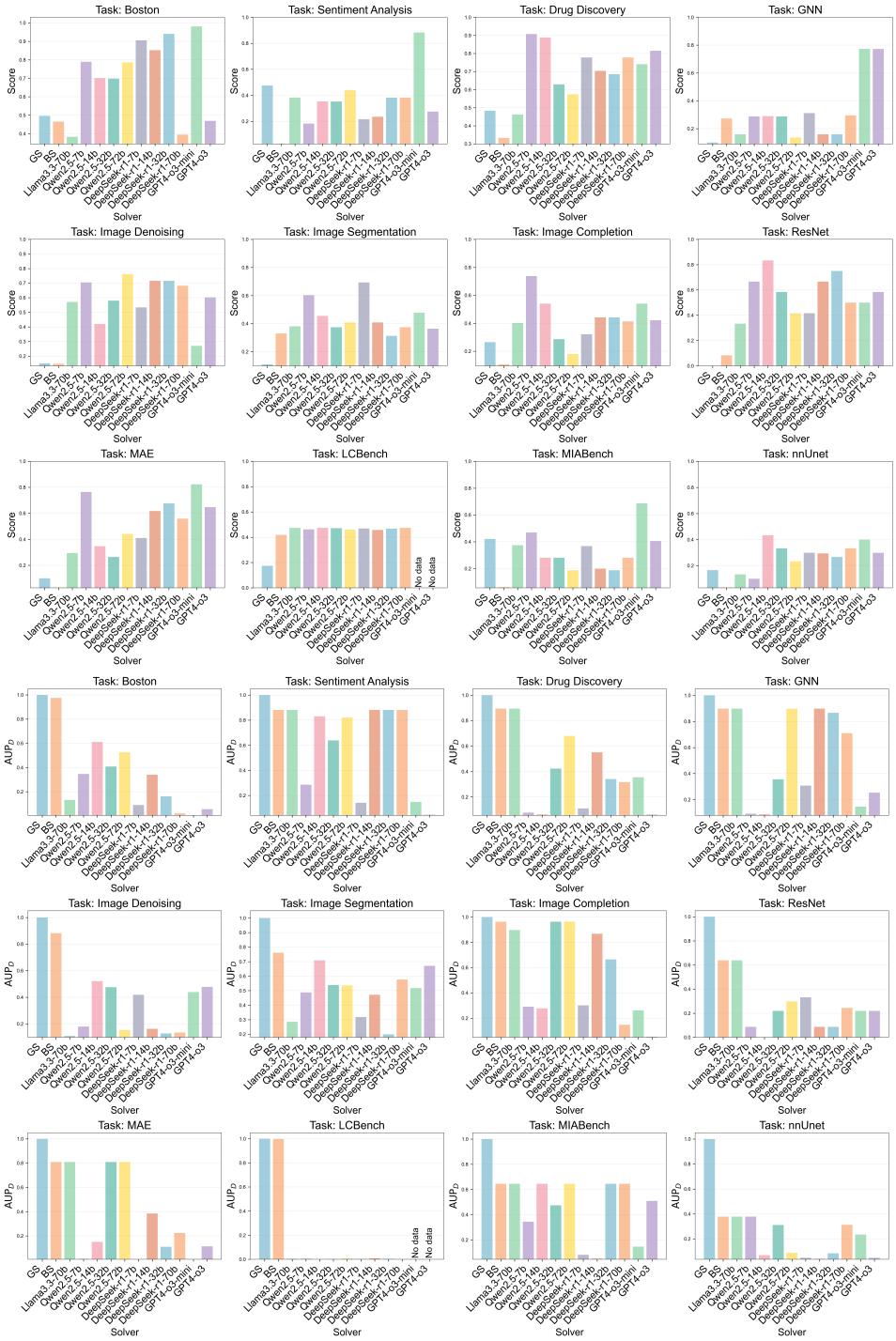

Figure 4: The top three rows present the Score of all solvers across different tasks, combining endpoint quality with discovery and stopping time. The bottom three rows present AUP$_D$, which evaluates the performance diversity of observed objective values and reflects their efficiency–diversity trade-off, independent of trial order. The two quantities are interpreted jointly.

tinue with little improvement. These patterns help interpret the joint endpoint and trajectory metrics. Across backbones, model size does not yield a monotonic improvement in this task (Supplementary Sect. D).

Result summary Across 12 heterogeneous tasks and language-model backbones (Supplementary Sect. B), these results demonstrate that SelfAI yields consistent trajectory-level benefits. SelfAI does not merely improve convergence accuracy but fundamentally reorganizes exploration dynamics in long-horizon scientific discovery. By reallocating experimental effort away from diminishing-return regions toward structurally informative areas of the search space, SelfAI implements a policy-adaptive exploration strategy that aligns with identifiable principles of long-horizon scientific discovery.

D LIMITATIONS

1) Context limitations. LCBench contains approximately 2000 trials, constituting a large and highly sparse search space. Performance depends on interacting hyperparameters such as weight decay, hidden-layer width, and depth, and high-performing regions are sharply localized. In practice, such large and sparse search spaces are common in scientific discovery tasks and can exceed the maximum context length supported by current GPT-family models.

2) **Computational fragility.** Table 3 compares the best result hit rates across solvers. The performance of LLM-based solvers is highly sensitive to the reasoning strategy, and early stopping often leads to a decline in hit rate. In the supplementary tables, we report the best result t_{best} for each solver. A run that does not find the optimal result is assigned $t_{\text{best}} = 1$. Moreover, when $t_{\text{best}} = 1$, the Score decreases substantially, highlighting the solver's weakness in stopping. As illustrated in Fig. 8, computational fragility also hinders reliable performance improvement through parameter tuning, as seen in the non-monotonic performance of the DeepSeek-R1 model series.

3) **Insufficient trajectory integration.** Supplementary Fig. 8 provides representative trajectory visualizations that highlight differences in evidence integration and stopping behavior. The Qwen2.5 models typically identify high-performing configurations early, with Qwen2.5-7b reaching near-optimal regions within the first 10-15 trials, followed by stable refinement with limited oscillation. In contrast, DeepSeek-R1 trajectories are more frequently non-monotonic. DeepSeek-R1-7b often oscillates between high- and low-performing regions, indicating instability in how accumulated evidence is leveraged, while in the tensor decomposition task DeepSeek-R1-70b can terminate prematurely after early improvements, leaving promising regions under-refined. Classical baselines (GS and BS) may also reach high-performing regions, but their improvements tend to be slower and less punctuated by rapid breakthroughs. Overall, these examples complement the aggregate metrics in the main text by illustrating how limitations can manifest as reduced trajectory stability, delayed breakthroughs, or suboptimal stopping decisions.

E ETHICAL CONSIDERATIONS

SelfAI is designed to assist scientific discovery and does not directly interact with sensitive personal data. However, potential risks include misuse in automated experimentation or over-reliance on AI-generated hypotheses without expert validation. We emphasize that human oversight remains essential, particularly in high-stakes domains such as medicine and drug discovery.

F BROADER IMPACTS

SelfAI has the potential to accelerate scientific discovery by improving the efficiency of experimentation and reducing redundant trials. This may benefit domains such as healthcare, materials science, and machine learning. However, it may also lower the barrier to large-scale automated experimentation, raising concerns about resource usage and unintended consequences. Responsible deployment and human oversight are essential.

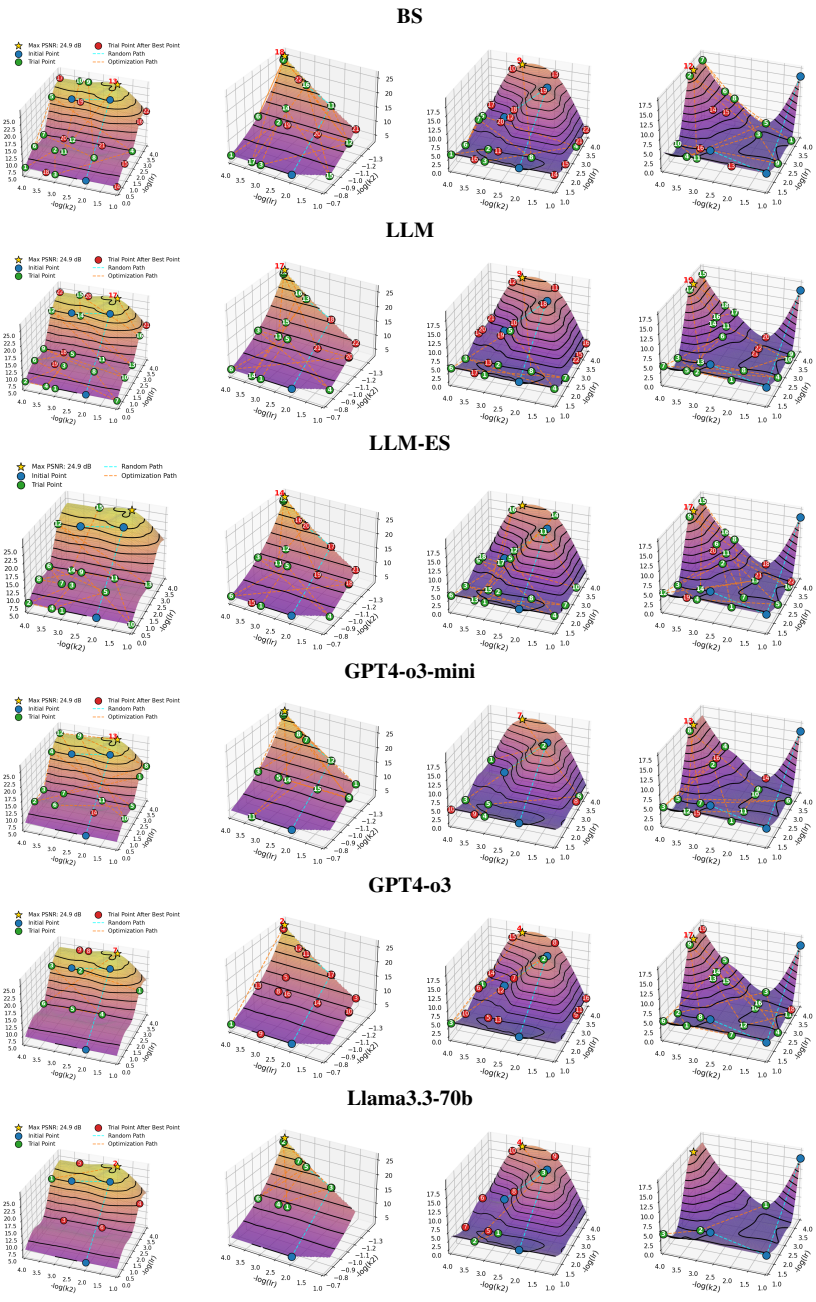

Figure 5: Illustration of the optimized trajectory using the SIREN method for additional cases in image denoising (first two columns) and segmentation (last two columns). Blue points are initial points. Green points represent suggested points before reaching the optimum. Red points indicate redundant suggestions generated after the optimal point has been reached. The $\star$ marks the optimal point. The numbered labels indicate the sequence of recommendations provided by LLMs.

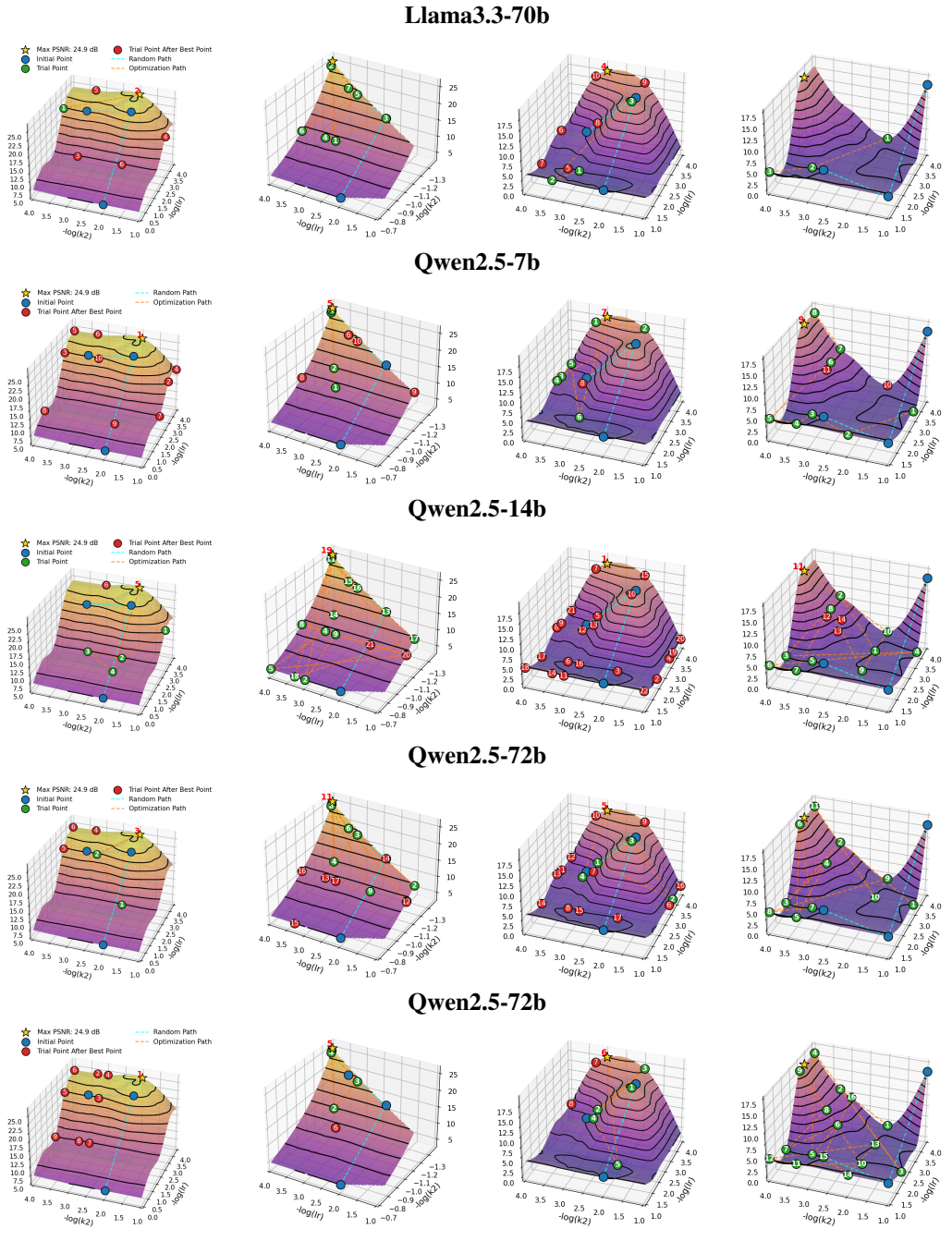

Figure 6: More optimized trajectories of Fig. 5.

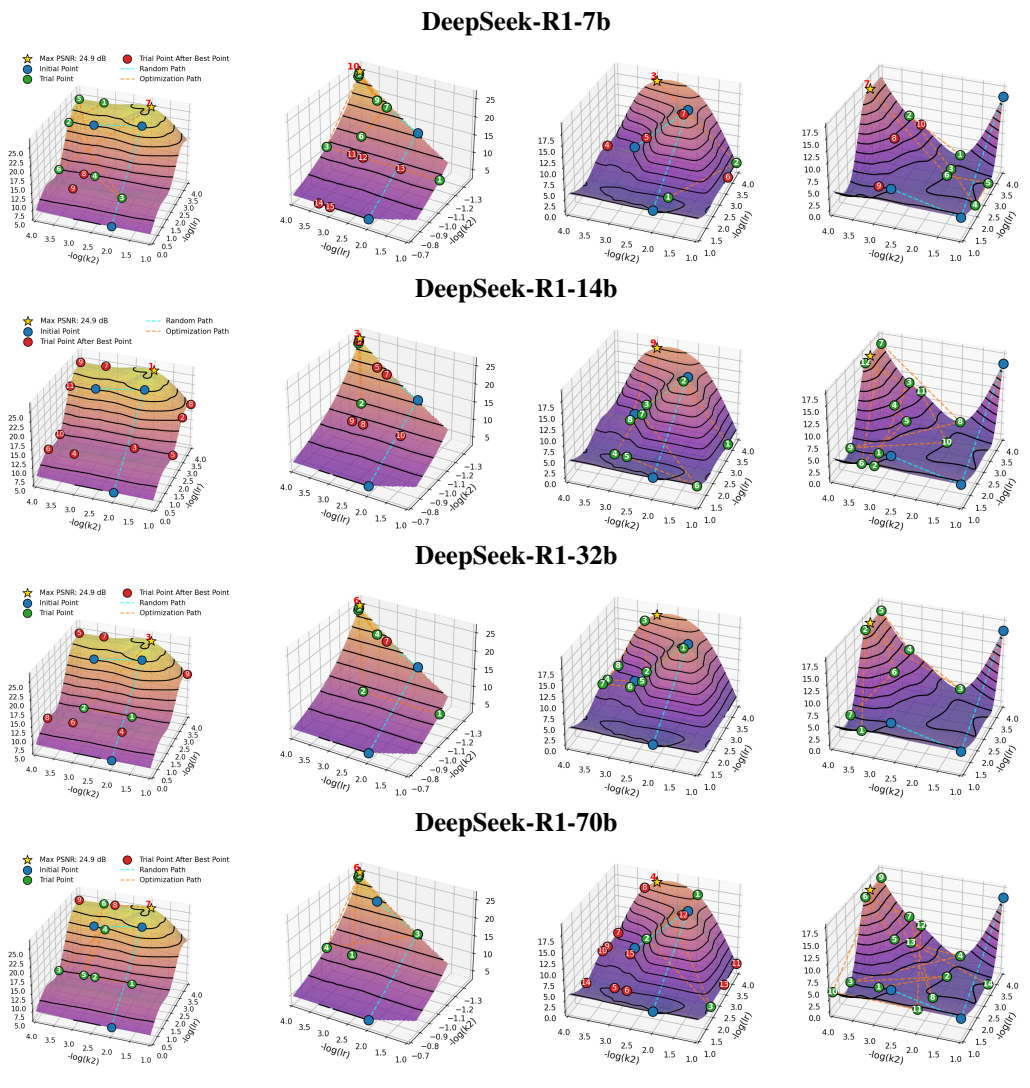

Figure 7: More optimized trajectories of Fig. 5. These LLMs have different suggestions for non-convex hyperparameter optimization.

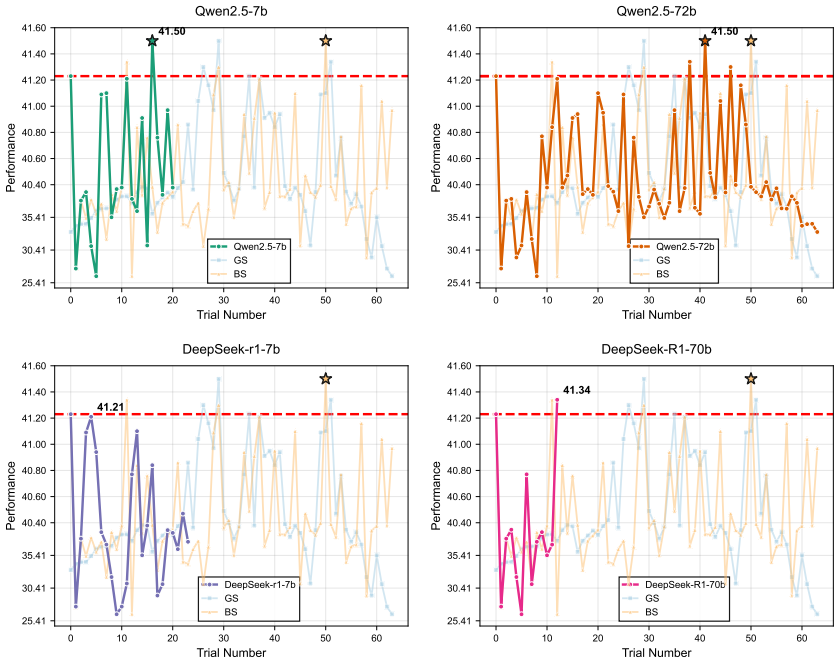

Figure 8: Failure Cases for the DeepSeek-R1 Family in the scientific computing field. Both DeepSeek-R1-7b and DeepSeek-R1-70b fail to reach the optimal results, marked with a star symbol ($\star$). Notably, DeepSeek-R1-7b fails to improve beyond the initial results, as indicated by the red dotted line, suggesting limited exploration capability in the search space.

G SUPPLEMENTARY TABLES

Table 9: Performance comparison of different solvers for the Boston housing prices based on the Random Forest Regressor.

Solver	Score↑	AUP$_D$ ↓	t_{best} ↓	t_{stop} ↓	Best Result↑	Rank
GS	0.4969	1.0000	0.0062	1.0000	0.841	9
BS	0.4654	0.9745	0.0755	0.9937	0.841	11
Llama3.3-70b	0.3831	0.1330	1.0000	0.1509	0.837	13
Qwen2.5-7b	0.7893	0.3463	0.0943	0.3270	0.841	5
Qwen2.5-14b	0.7013	0.6108	0.0063	0.5912	0.841	7
Qwen2.5-32b	0.6981	0.4081	0.2579	0.3459	0.841	8
Qwen2.5-72b	0.7862	0.5250	0.0126	0.4151	0.841	6
DeepSeek-R1-7b	0.9057	0.0910	0.0881	0.1006	0.841	3
DeepSeek-R1-14b	0.8522	0.3401	0.0063	0.2893	0.841	4
DeepSeek-R1-32b	0.9403	0.1618	0.0063	0.1132	0.841	2
DeepSeek-R1-70b	0.3948	0.0215	1.0000	0.0189	0.833	12
GPT4-o3-mini	0.9811	0.0062	0.0189	0.0189	0.841	1
GPT4-o3	0.4694	0.0560	1.0000	0.0377	0.840	10

Table 10: Performance comparison of different solvers for the sentiment analysis task based on the LSTM model.

Solver	Score↑	AUP$_D$ ↓	t_{best} ↓	t_{stop} ↓	Best Result↑	Rank
GS	0.4750	1.0000	0.0500	1.0000	0.96	2
BS	0.0294	0.8817	0.9412	1.0000	0.96	13
Llama3.3-70b	0.3824	0.8817	0.2353	1.0000	0.96	4
Qwen2.5-7b	0.1830	0.2857	1.0000	0.5294	0.94	12
Qwen2.5-14b	0.3529	0.8318	0.4118	0.8824	0.96	7
Qwen2.5-32b	0.3529	0.6383	0.5882	0.7059	0.96	7
Qwen2.5-72b	0.4412	0.8220	0.2353	0.8824	0.96	3
DeepSeek-R1-7b	0.2157	0.1419	1.0000	0.3529	0.93	11
DeepSeek-R1-14b	0.2353	0.8817	0.5294	1.0000	0.96	10
DeepSeek-R1-32b	0.3824	0.8817	0.2353	1.0000	0.96	4
DeepSeek-R1-70b	0.3824	0.8817	0.2353	1.0000	0.96	4
GPT4-o3-mini	0.8824	0.1494	0.0588	0.1765	0.96	1
GPT4-o3	0.2745	0.0469	1.0000	0.1765	0.93	9

Table 11: Performance comparison of different solvers for scientific computing fields, where we evaluate the tensor wheel decomposition method on the multi-spectral image completion dataset.

Solver	Score↑	AUP$_D$ ↓	t_{best} ↓	t_{stop} ↓	Best Result↑	Rank
GS	0.2656	1.0000	0.4688	1.0000	41.50	11
BS	0.1066	0.9638	0.7869	1.0000	41.50	13
Llama3.3-70b	0.4016	0.8979	0.2623	0.9344	41.50	8
Qwen2.5-7b	0.7377	0.2904	0.2295	0.2951	41.50	1
Qwen2.5-14b	0.5410	0.2779	0.4262	0.4918	41.50	2
Qwen2.5-32b	0.2869	0.9638	0.4262	1.0000	41.50	10
Qwen2.5-72b	0.1803	0.9638	0.6393	1.0000	41.50	12
DeepSeek-R1-7b	0.3216	0.3009	1.0000	0.3443	41.21	9
DeepSeek-R1-14b	0.4426	0.8687	0.2295	0.8852	41.50	4
DeepSeek-R1-32b	0.4426	0.6645	0.4426	0.6721	41.50	4
DeepSeek-R1-70b	0.4136	0.1477	1.0000	0.1639	41.34	7
GPT4-o3-mini	0.5410	0.2623	0.4590	0.4590	41.50	2
GPT4-o3	0.4217	0.0539	1.0000	0.1475	41.34	6

Table 12: Performance comparison of different solvers for the image segmentation task based on the SIREN model.

Solver	Score↑	AUP$_D$ ↓	t_{best} ↓	t_{stop} ↓	Best Result↑	Rank
GS	0.1100	1.0000	0.7800	1.0000	16.63	13
BS	0.3295	0.7612	0.4773	0.8636	16.63	11
Qwen2.5-7b	0.6023	0.4876	0.3636	0.4318	16.63	2
Qwen2.5-14b	0.4545	0.7085	0.2727	0.8182	16.63	4
Qwen2.5-32b	0.3743	0.5382	0.6136	0.6364	16.60	9
Qwen2.5-72b	0.4088	0.5362	0.6364	0.5455	16.60	5
DeepSeek-R1-7b	0.6932	0.3187	0.2273	0.3864	16.63	1
DeepSeek-R1-14b	0.4085	0.4718	0.7045	0.4773	16.60	6
DeepSeek-R1-32b	0.3125	0.1984	1.0000	0.3409	15.77	12
DeepSeek-R1-70b	0.3746	0.5763	0.5909	0.6591	16.60	8
Llama3.3-70b	0.3803	0.2859	0.5909	0.2955	10.79	7
GPT4-o3-mini	0.4773	0.5177	0.4545	0.5909	16.63	3
GPT4-o3	0.3636	0.6714	0.4773	0.7955	16.63	10

Table 13: Performance comparison of different solvers for the image denoising task based on the SIREN model.

Solver	Score↑	AUP$_D$ ↓	t_{best} ↓	t_{stop} ↓	Best Result↑	Rank
GS	0.1500	1.0000	0.7000	1.0000	24.78	12
BS	0.1477	0.8819	0.7045	1.0000	24.78	13
Qwen2.5-7b	0.7045	0.1802	0.1364	0.4545	24.78	4
Llama3.3-70b	0.5706	0.1105	0.5455	0.2955	24.27	8
Qwen2.5-14b	0.4205	0.5205	0.5455	0.6136	24.78	10
Qwen2.5-32b	0.5795	0.4762	0.3182	0.5227	24.78	7
Qwen2.5-72b	0.7614	0.1543	0.1364	0.3409	24.78	1
DeepSeek-R1-7b	0.5341	0.4193	0.3864	0.5455	24.78	9
DeepSeek-R1-14b	0.7159	0.1639	0.0909	0.4773	24.78	2
DeepSeek-R1-32b	0.7159	0.1284	0.2045	0.3636	24.78	2
DeepSeek-R1-70b	0.6818	0.1359	0.2955	0.3409	24.78	5
GPT4-o3-mini	0.2703	0.4389	0.7955	0.6591	24.48	11
GPT4-o3	0.6023	0.4786	0.2045	0.5909	24.78	6

Table 14: Performance comparison of different solvers for Masked Autoencoders (MAE) (He et al., 2022).

Solver	Score↑	AUP$_D$ ↓	t_{best} ↓	t_{stop} ↓	Best Result↑	Rank
GS	0.1000	1.0000	0.8000	1.0000	85.0	12
BS	0.0294	0.8093	0.9412	1.0000	85.0	13
Llama3.3-70b	0.2941	0.8093	0.4118	1.0000	85.0	10
Qwen2.5-7b	0.7647	0.0112	0.2353	0.2353	85.0	2
Qwen2.5-14b	0.3471	0.1513	1.0000	0.2941	84.5	9
Qwen2.5-32b	0.2647	0.8093	0.4706	1.0000	85.0	11
Qwen2.5-72b	0.4412	0.8093	0.1176	1.0000	85.0	7
DeepSeek-R1-7b	0.4104	0.0080	1.0000	0.1765	84.9	8
DeepSeek-R1-14b	0.6176	0.3853	0.2941	0.4706	85.0	5
DeepSeek-R1-32b	0.6765	0.1108	0.2941	0.3529	85.0	3
DeepSeek-R1-70b	0.5588	0.2235	0.1765	0.7059	85.0	6
GPT4-o3-mini	0.8235	0.0088	0.1176	0.2353	85.0	1
GPT4-o3	0.6471	0.1141	0.3529	0.3529	85.0	4

Table 15: Performance comparison of different solvers for the image classification task on the ImageNet dataset reported from the ResNet benchmark.

Solver	Score↑	$\text{AUP}_D \downarrow$	$t_{\text{best}} \downarrow$	$t_{\text{stop}} \downarrow$	Best Result↑	Rank
GS	0.0000	1.0000	1.0000	1.0000	21.43	13
BS	0.0833	0.6394	0.8333	1.0000	21.43	12
Llama3.3-70b	0.3333	0.6394	0.3333	1.0000	21.43	11
Qwen2.5-7b	0.6667	0.0865	0.3333	0.3333	21.43	3
Qwen2.5-14b	0.8333	0.0000	0.1667	0.1667	21.43	1
Qwen2.5-32b	0.5833	0.2192	0.3333	0.5000	21.43	5
Qwen2.5-72b	0.4167	0.2988	0.5000	0.6667	21.43	9
DeepSeek-R1-7b	0.4167	0.3322	0.5000	0.6667	21.43	9
DeepSeek-R1-14b	0.6667	0.0865	0.3333	0.3333	21.43	3
DeepSeek-R1-32b	0.7500	0.0865	0.1667	0.3333	21.43	2
DeepSeek-R1-70b	0.5000	0.2443	0.5000	0.5000	21.43	7
GPT4-o3-mini	0.5000	0.2192	0.5000	0.5000	21.43	7
GPT4-o3	0.5833	0.2192	0.3333	0.5000	21.43	5

Table 16: Performance comparison of different solvers for the image classification task on the LCBench dataset reported from Zimmer et al. (2021).

Solver	Score↑	$\text{AUP}_D \downarrow$	$t_{\text{best}} \downarrow$	$t_{\text{stop}} \downarrow$	Best Result↑	Rank
GS	0.1750	1.0000	0.6500	1.0000	88.29	11
BS	0.4189	0.9985	0.1622	1.0000	88.29	10
Llama3.3-70b	0.4738	0.0052	1.0000	0.0486	87.98	3
Qwen2.5-7b	0.4619	0.0047	1.0000	0.0451	85.78	7
Qwen2.5-14b	0.4739	0.0017	1.0000	0.0180	85.62	1
Qwen2.5-32b	0.4718	0.0022	1.0000	0.0225	85.62	4
Qwen2.5-72b	0.4599	0.0081	1.0000	0.0766	87.98	8
DeepSeek-R1-7b	0.4691	0.0027	1.0000	0.0280	85.62	5
DeepSeek-R1-14b	0.4579	0.0086	1.0000	0.0806	87.98	9
DeepSeek-R1-32b	0.4677	0.0035	1.0000	0.0331	85.78	6
DeepSeek-R1-70b	0.4739	0.0014	1.0000	0.0180	85.62	1

Table 17: Performance comparison of different solvers for the medical image segmentation task on the BTCV dataset reported from the nnUnet benchmark Isensee et al. (2024).

Solver	Score↑	AUP$_D$ ↓	t_{best} ↓	t_{stop} ↓	Best Result↑	Rank
GS	0.4211	1.0000	0.1579	1.0000	85.04	3
BS	0.0625	0.6453	0.8750	1.0000	85.04	13
Llama3.3-70b	0.3750	0.6453	0.2500	1.0000	85.04	5
Qwen2.5-7b	0.4688	0.3441	0.5000	0.5625	85.04	2
Qwen2.5-14b	0.2813	0.6453	0.4375	1.0000	85.04	7
Qwen2.5-32b	0.2813	0.4742	0.6875	0.7500	85.04	7
Qwen2.5-72b	0.1875	0.6453	0.6250	1.0000	85.04	11
DeepSeek-R1-7b	0.3680	0.0813	1.0000	0.1250	83.69	6
DeepSeek-R1-14b	0.1983	0.0505	1.0000	0.1875	80.69	10
DeepSeek-R1-32b	0.1875	0.6453	0.6250	1.0000	85.04	11
DeepSeek-R1-70b	0.2813	0.6453	0.4375	1.0000	85.04	7
GPT4-o3-mini	0.6875	0.1466	0.2500	0.3750	85.04	1
GPT4-o3	0.4063	0.5081	0.4375	0.7500	85.04	4

Table 18: Performance comparison of different solvers for the nnUnet model on the BraTS dataset LaBella et al. (2023).

Solver	Score↑	AUP$_D$ ↓	t_{best} ↓	t_{stop} ↓	Best Result↑	Rank
GS	0.1667	1.0000	0.6667	1.0000	82.45	10
BS	0.0333	0.3786	0.9333	1.0000	82.45	13
Llama3.3-70b	0.1333	0.3786	0.7333	1.0000	82.45	11
Qwen2.5-7b	0.1000	0.3786	0.8000	1.0000	82.45	12
Qwen2.5-14b	0.4333	0.0698	0.5333	0.6000	82.45	1
Qwen2.5-32b	0.3333	0.3118	0.5333	0.8000	82.45	3
Qwen2.5-72b	0.2333	0.0883	0.7333	0.8000	82.45	9
DeepSeek-R1-7b	0.2995	0.0481	1.0000	0.4000	82.41	5
DeepSeek-R1-14b	0.2940	0.0406	1.0000	0.4000	82.00	7
DeepSeek-R1-32b	0.2667	0.0839	0.6667	0.8000	82.45	8
DeepSeek-R1-70b	0.3333	0.3126	0.5333	0.8000	82.45	3
GPT4-o3-mini	0.4000	0.2347	0.6000	0.6000	82.45	2
GPT4-o3	0.2984	0.0484	1.0000	0.4000	82.33	6

Table 19: Performance comparison of different solvers for the GraphSAGE model on the imbalanced node classification task.

Solver	Score↑	AUP$_D$ ↓	t_{best} ↓	t_{stop} ↓	Best Result↑	Rank
GS	0.1000	1.0000	0.8000	1.0000	89.28	13
BS	0.2727	0.8982	0.4545	1.0000	89.28	8
Llama3.3-70b	0.1591	0.8982	0.6818	1.0000	89.28	9
Qwen2.5-7b	0.2880	0.0928	1.0000	0.4091	89.12	6
Qwen2.5-14b	0.2894	0.0890	1.0000	0.4091	89.15	5
Qwen2.5-32b	0.2880	0.3564	1.0000	0.4091	89.12	6
Qwen2.5-72b	0.1364	0.8982	0.7273	1.0000	89.28	12
DeepSeek-R1-7b	0.3116	0.3083	1.0000	0.3636	89.15	3
DeepSeek-R1-14b	0.1591	0.8982	0.6818	1.0000	89.28	9
DeepSeek-R1-32b	0.1591	0.8657	0.7273	0.9545	89.28	9
DeepSeek-R1-70b	0.2955	0.7096	0.5909	0.8182	89.28	4
GPT4-o3-mini	0.7727	0.1463	0.0455	0.4091	89.28	1
GPT4-o3	0.7727	0.2540	0.1818	0.2727	89.28	1

Table 20: Performance comparison of different solvers for bioactivity prediction on Chagas EP20 dataset Ekins et al. (2015)

Solver	Score↑	AUP$_D$ ↓	t_{best} ↓	t_{stop} ↓	Best Result↑	Rank
GS	0.4833	1.0000	0.0333	1.0000	0.754	11
BS	0.3333	0.8941	0.3333	1.0000	0.754	13
Llama3.3-70b	0.4630	0.8941	0.0741	1.0000	0.754	12
Qwen2.5-7b	0.9074	0.0772	0.0741	0.1111	0.754	1
Qwen2.5-14b	0.8889	0.0649	0.1111	0.1111	0.754	2
Qwen2.5-32b	0.6296	0.4231	0.0741	0.6667	0.754	9
Qwen2.5-72b	0.5741	0.6799	0.0741	0.7778	0.754	10
DeepSeek-R1-7b	0.7778	0.1104	0.2222	0.2222	0.754	4
DeepSeek-R1-14b	0.7037	0.5510	0.0741	0.5185	0.754	7
DeepSeek-R1-32b	0.6852	0.3405	0.0741	0.5556	0.754	8
DeepSeek-R1-70b	0.7778	0.3160	0.1111	0.3333	0.754	4
GPT4-o3-mini	0.7407	0.3546	0.1852	0.3333	0.754	6
GPT4-o3	0.8148	0.0622	0.0370	0.3333	0.754	3

H PROMPTS OF COMPARED METHODS

Prompt of LLM solver

Instructions:
Search Space (Numbering starts from 0, excluding Completed Trials): `{{TRIALS}}`
Task 1: Optimization Recommendation Recommend exactly `{{N_JOBS}}` promising trials from the provided **Search Space** (include both number and params).
Rules:
1. "params" MUST include:
`{{HYPERNAME}}`
2. All selected 'params' must match exactly with the provided **Search Space**. Do NOT leave out any key.
3. Use the analysis in **Task 1** (trial analysis, performance trends, highlights, and other insights) to guide selection.
4. Based on the above analysis, explore under-explored regions only when there is clear evidence of potential performance gain.
5. Do not mix, modify, or create new values.
6. You MUST not output any JSON blocks in this part.
7. You MUST provide reasoning for each recommendation.

Early-Stopping Prompt of LLM-ES solver

Completed trials: `{{COMPLETED_TRIALS}}`
The following **Search Space** contains **unexplored** trials. `{{TRIALS}}`
If the optimization process should be stopped, Answer: Yes with confidence score: `{{CONFIDENCE_SCORE}}`. Otherwise, Answer: No with confidence score: `{{CONFIDENCE_SCORE}}`.
Finally, you MUST output 'Answer: No/Yes' with confidence score: `{{CONFIDENCE_SCORE}}`.