NATUREBENCH: CAN CODING AGENTS MATCH THE PUBLISHED SOTA OF NATURE-FAMILY PAPERS?

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce **NatureBench**, a cross-discipline benchmark of 90 tasks distilled from peer-reviewed Nature-family publications, designed to evaluate whether AI coding agents can move beyond *reproduction* toward *discovery* on real scientific problems. NatureBench is built on **NatureGym**, an automated pipeline that constructs a standardized, per-task containerized environment from a source paper, addressing the environment-fragmentation problem that has limited the credibility of prior agent-on-research benchmarks. Evaluating sixteen frontier agent configurations under a strict web-search-disabled protocol, we find that the strongest model surpasses SOTA on only 23.3% of tasks under the $g > 0.1$ criterion. Analysis of method pathways reveals that agents succeed primarily through methodological translation, converting scientific tasks into familiar supervised prediction problems, rather than through genuine scientific invention. Failures are dominated by wrong method choice and judge-identified shortcuts, followed by insufficient compute budget, not by task misunderstanding. We release the benchmark, the NatureGym pipeline, and a public leaderboard with maintainer-side reproduction.

1 INTRODUCTION

AI coding agents are rapidly moving toward autonomous scientific research (Karpathy, 2026; Lu et al., 2026; Gottweis et al., 2026), from reproducing published implementations to conducting end-to-end research workflows. As these systems begin to target real scientific problems, rigorous evaluation becomes critical: without reliable benchmarks, it is impossible to tell whether an agent is genuinely advancing the state of the art or merely fitting familiar patterns to new data.

However, existing benchmarks for evaluating agent capabilities on scientific research have several limitations (see Appendix A). Paper-based benchmarks (Starace et al., 2025; Siegel et al., 2024; Wang et al., 2026) measure whether an agent can re-implement a published method, but stop short of the more consequential question: can an agent *discover* a competitive method on its own? Engineering-optimization benchmarks (Chan et al., 2025; Rank et al., 2026; Nathani et al., 2025; Qiang et al., 2026) target Kaggle competitions or post-training tasks, which do not require the domain reasoning, specialized tooling, or cross-discipline knowledge that characterize research in the natural sciences, and suffer from environment fragmentation that makes independent re-running fragile. Credibly evaluating whether autonomous research agents can advance the frontier of AI-for-Science requires a benchmark that is both challenging and bidirectional. It must test *discovery*, whether an agent can devise methods that surpass the published state of the art, on *genuine scientific problems* drawn from the natural sciences rather than on engineering proxies.

We present **NatureBench**, a cross-discipline benchmark of 90 tasks distilled from peer-reviewed Nature-family publications, designed to evaluate whether AI coding agents can move beyond reproduction toward discovery. NatureBench simultaneously extends both axes: the **PaperBench axis** from Understanding $\rightarrow$ Coding to Discovery, and the **PostTrainBench axis** from Engineering Optimization to Science. It is built on **NatureGym**, an automated pipeline that converts a paper into a containerized task package comprising a task brief, the paper's dataset, a held-out test set with hidden ground truth, and an automated evaluator, addressing the environment-fragmentation problem in prior benchmarks. We collect approximately $5{,}500$ papers from ten Nature-family journals published between 2022 and 2025 and apply a three-stage build-then-verify pipeline to yield the final 90 task packages (Figure 2). An information firewall removes the source method from each package,

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.

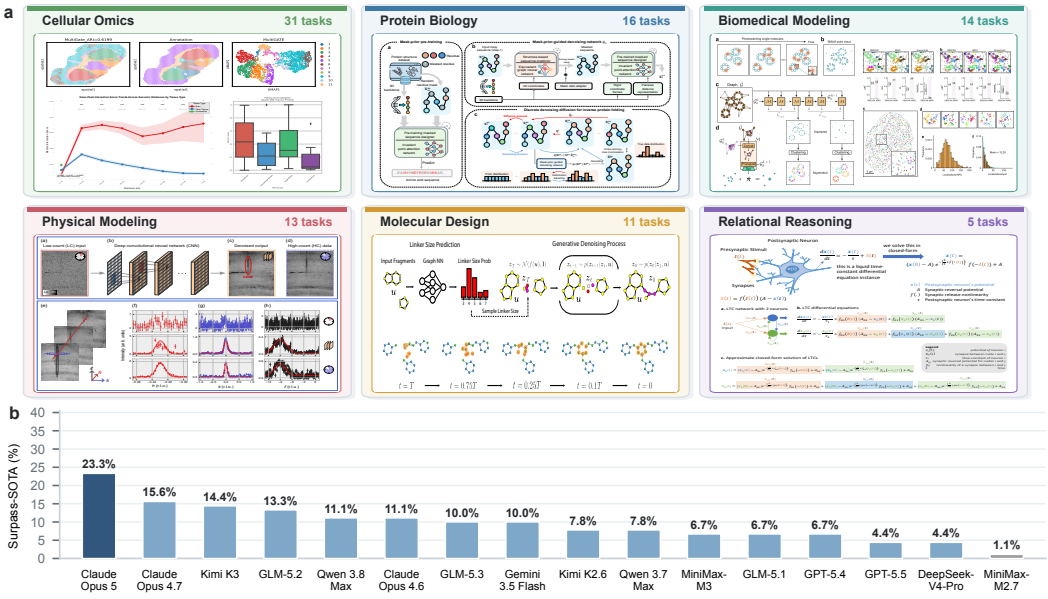

Figure 1: **NatureBench overview.** (a) Six task domains with representative source figures (Miao et al., 2025; Bai et al., 2025; Pineda et al., 2025; Oppliger et al., 2024; Igashov et al., 2024; Hasani et al., 2022). (b) NatureBench leaderboard by Surpass-SOTA ($g > 0.1$) across sixteen models.

so agents must discover solutions rather than reproduce them. The benchmark spans six scientific task domains and uses a SOTA-normalized relative gap g as the metric, supplemented by a post-hoc validity judge that detects shortcut behaviors such as output fabrication and feedback gaming.

We evaluate sixteen agents spanning three coding-agent harnesses (Claude Code, Codex CLI, Gemini CLI) and sixteen frontier models under a web-search-disabled protocol, as shown in Figure 1. The strongest agent, Claude Opus 5, surpasses the published SOTA ($g > 0.1$) on only 23.3% of tasks and matches it on 57.8%. A sixteen-agent behavioral analysis over $1{,}440$ task–agent runs reveals that success is driven primarily by methodological translation, where agents convert scientific tasks into familiar supervised-prediction problems, accounting for 45.9% of validated successes, rather than by scientific invention. Failures are dominated by wrong method choice (29.8%) and judge-identified shortcuts (25.5%), followed by insufficient compute budget (17.6%), not by task misunderstanding. Our contributions are as follows:

- **NatureGym**, an automated pipeline that constructs reproducible, containerized per-task environments from Nature-family papers, addressing the environment-fragmentation problem that has limited the credibility of prior agent-on-research benchmarks.
- **NatureBench**, a benchmark of 90 Nature-sourced tasks across six scientific task domains with a Discovery-oriented evaluation protocol (Surpass-SOTA, Match-SOTA, validity judge) that separates genuine algorithmic progress from engineering optimization and shortcut-taking.

2 NATUREGYM

We introduce **NatureGym**, a pipeline that turns a published Nature-family paper into a ready-to-run agentic task. Each task is a containerized package comprising a task brief, the dataset, a held-out test set, an automated evaluator, and a SOTA anchor score. NatureGym standardizes papers with heterogeneous formats, toolchains, and data modalities into one reproducible task format, while imposing an information firewall that withholds the original method so that agents must discover solutions rather than reproduce them.

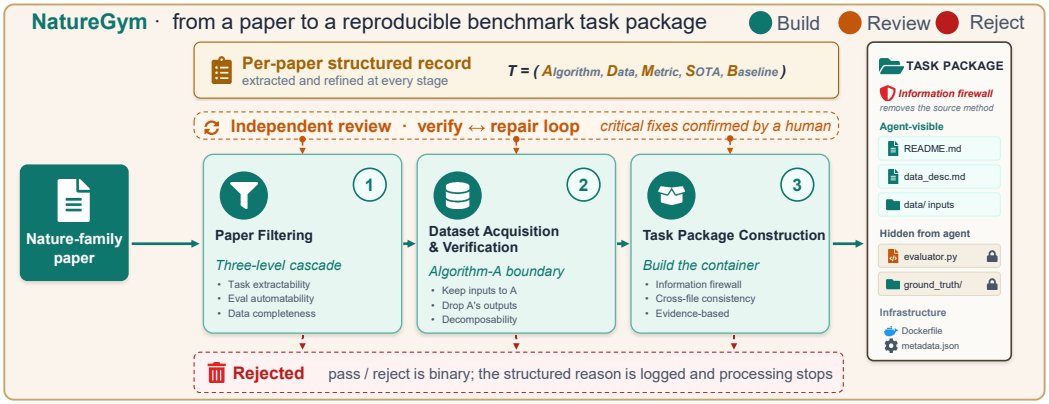

Figure 2: **The NatureGym pipeline.** Three review-gated stages turn one Nature-family paper into a containerized task package, refining a shared per-paper record $T = (A, D, M, S, B)$ along the way.

2.1 Pipeline Overview

As shown in Figure 2, NatureGym builds each task through three stages: Paper Filtering (§2.2), Dataset Acquisition and Verification (§2.3), and Task Package Construction (§2.4). Each stage ends with an independent review through a verify–repair loop before the next stage begins.

Each stage makes a pass-or-reject decision, and extracts and refines task information in a per-paper record; rejected papers undergo no further processing. Initialized during the filtering stage, the record $T = (A, D, M, S, B)$ contains a core algorithm A, dataset D, metric M, SOTA score S, and optional baseline B. Task construction uses the accumulated record without re-reading the paper. Every stage is run by an LLM agent, and a human confirms the critical corrections that each review surfaces. Appendix B details each stage of the NatureGym pipeline.

2.2 Paper Filtering

Paper filtering identifies candidate papers suitable for task construction through three steps: preprocessing, a three-level cascade filter, and an adversarial review.

- **Preprocessing.** We retain only research articles and convert each into markdown text, full-page screenshots of figures and tables, and a categorized hyperlink list for subsequent filtering.
- **Three-level filtering.** We assess three feasibility dimensions. **Level 1: Task.** The core contribution must yield an extractable ML task, excluding auxiliary uses of ML, non-computational studies, and tasks requiring physical interaction. **Level 2: Evaluation.** The paper must claim SOTA on a quality-related metric. Evaluation must be deterministic and fully automated. **Level 3: Data.** At least one evaluation instance—development data (D_{dev}), evaluation data (D_{eval}), and metrics—must have complete, publicly accessible data matching the paper's version. We also tag each dataset by volume and reject papers whose data exceeds 50 GB.
- **Filtering review.** Before entering the costly data-acquisition stage, a separate adversarial pass re-examines every paper that passed, targeting false positives. It rechecks both the pass-or-reject decision and the extracted task information, writing corrections back into the per-paper record. Critical overrides are confirmed by a human.

2.3 Dataset Acquisition and Verification

Papers that pass filtering enter dataset acquisition, where we download the data, determine the boundary separating the task definition from the paper's core algorithm, and re-verify data completeness against the actual files rather than the metadata-level probes of the filtering stage.

- **Dataset acquisition.** We clone the linked code and data repositories and download the datasets by size tier and priority, taking the evaluation instances behind the paper's main results first. Downloads are bounded by the size-tier rules established during filtering.

- **File-level firewall.** The agent must start exactly where the core algorithm A starts, so it receives the inputs to A but none of A's operations or outputs. Files that define the task or provide legitimate initial inputs and resources are retained; files containing A's own operations or outputs, as well as irrelevant files, are excluded. We make each decision by reading the paper, the code, and the materialized data together.
- **Dataset verification and review.** Verification on the downloaded files focuses on two key properties, decomposability and instance validity: whether D_{dev} separates from D_{eval} and X_{test} from Y_{ref}, and whether the retained instances correspond to a single research objective and include the core experiment. At least one instance must be complete. We record the split procedure here and perform the actual partitioning in §2.4. A separate read-only review rechecks the A-boundary and recorded descriptions against the paper, code, and files; subsequent corrections update the record and data, with extensive corrections confirmed manually.

2.4 Task Package Construction

Each paper that passes filtering and data verification is assembled into a task package, as shown in Figure 2. Construction and subsequent verification follow three principles: (i) **Evidence-grounded fidelity**: every component and performance anchor must be supported by verified records and source evidence. (ii) **Information firewall**: no file may reveal the source paper's identity or method, and task inputs must be separated from hidden references and scoring logic. (iii) **Executable integrity**: all components must be mutually consistent in semantics and interfaces, and the package as a whole must pass both static checks and end-to-end execution.

- **Data organization and task documentation.** Following the decomposition procedure from §2.3, we separate agent-visible inputs from hidden references according to the reference-answer type (static label, oracle function, or distributional statistic). Two task documents define the data, task, evaluation metrics, and submission specification. Metadata records the scientific domain, compute requirements, and per-instance SOTA scores.
- **Automated evaluator and execution environment.** The evaluator validates and independently scores agent outputs according to the reference-answer type, isolating failures across instances. A shared base image pre-installs core scientific and ML libraries, with task-specific dependencies layered on top via per-task Dockerfiles and standalone builds reserved for irreconcilable conflicts.
- **Package and environment review.** Unlike the one-shot reviews of previous stages, this review follows an iterative verify–repair loop with three stages: a build-time self-audit, a separate task-package verification comprising 36 automated checks, and an environment verification through Docker builds and library smoke tests on a physical machine. Failed checks trigger minimal targeted repairs and immediate re-verification; failed automated repairs go to human review.

3 NatureBench

NatureBench comprises 90 tasks spanning six scientific domains, produced by applying NatureGym (§2) to Nature-family journal papers. It is the first to combine paper-sourced tasks, genuine scientific problems, and optimization-oriented evaluation against published SOTA (Table 1).

3.1 Source Corpus

We define the source pool by journal and publication year, then apply the NatureGym pipeline.

Journal selection. We select ten Nature-family journals whose papers include concrete algorithmic contributions with numerical SOTA claims and publicly available data, and whose topical scope covers scientific machine learning. The publication window is 2022–2025, chosen to balance corpus size against software-stack currency and data-contamination risk. See Appendix C for details.

Pipeline funnel. All collected papers pass through five phases: the three NatureGym stages (§2) bookended by an initial collection crawl and a final calibration step. *Collection* crawls ∼5,500 initial candidates from ten Nature-family journals. *Filtering* retains ∼2,500 research articles via an article-type filter, then applies three-stage filtering (§2.2) to yield ∼200 papers. *Acquisition* acquires and

Table 1: **Comparison with representative benchmarks. # Tasks** reports the number of primary evaluation units. **Paper** indicates whether tasks are derived from source papers; **Science** indicates whether they address scientific domains beyond AI/ML methodology itself; and **Optimization** indicates whether agents maximize task performance rather than recover or assess known results.

Benchmark	Source	# Tasks	Paper	Science	Optimization	Objective	Scoring anchor
ML Paper Replication							
PaperBench (Starace et al., 2025)	ICML papers	20	✓	×	×	paper replication	author rubrics
AutoExperiment (Kim et al., 2025)	ML papers	85	✓	×	×	masked-code reproduction	gold outputs
FIRE-Bench (Wang et al., 2026)	LLM analysis papers	30	✓	×	×	finding rediscovery	paper claims
Scientific Paper Reproduction							
CORE-Bench (Siegel et al., 2024)	Code Ocean capsules	270	✓	✓	×	result reproduction	manual outputs
REPRO-Bench (Hu et al., 2025)	social-science papers	112	✓	✓	×	reproducibility assessment	expert labels
ReplicationBench (Ye et al., 2025)	astrophysics papers	111	✓	✓	×	result replication	reported values
AutoMat (Huang et al., 2026)	materials-science papers	85	✓	✓	×	claim reproduction	expert annotations
Collider-Bench (Faroughy et al., 2026)	LHC papers	10	✓	✓	×	analysis reproduction	event yields
Task-Performance Optimization							
MLE-bench (Chan et al., 2025)	Kaggle competitions	75	×	×	✓	ML engineering	Kaggle leaderboard
PostTrainBench (Rank et al., 2026)	model–benchmark pairs	28	×	×	✓	LLM post-training	official instruct models
MLS-Bench (Lyu et al., 2026a)	ML research problems	140	×	×	✓	method invention	human baselines
AutoLab (Xu et al., 2026)	expert-curated problems	36	×	×	✓	long-horizon optimization	baseline/human metrics
NatureBench (ours)	**Nature-family papers**	**90**	✓	✓	✓	**method development**	**published SOTA**

verifies datasets (§2.3), narrowing to ∼180. *Construction* builds and verifies task packages (§2.4), retaining ∼160. *Calibration* removes defective tasks via evaluation-time quality calibration (§3.2), finalizing the benchmark at 90 task packages. Table 4 reports counts at each step.

3.2 Benchmark Quality Calibration

Build-time verification (§2) guarantees only that a task package is structurally well-formed and runnable. Some defects surface only when an agent actually attempts to solve the task. We therefore add an evaluation-time quality calibration before the main experiments, proceeding in three steps. Appendix D provides full details.

First-round diagnosis and repair. We run Claude Opus 4.6 over all tasks in base mode and diagnose each case by combining the score, the agent trajectory, and the task package. Exposed defects include ground-truth leakage, distorted task definitions, metrics that fail to distinguish shortcuts from genuine solutions, evaluator inconsistencies, pipeline or environment errors, and missing data. Locally verifiable defects receive minimal repairs. Tasks with irreparable issues are dropped. Legitimate low scores are retained.

Reproduction-mode package audit. In reproduce mode, the agent additionally receives the source paper and is instructed to faithfully reproduce its method. We run Claude Opus 4.6 and DeepSeek-V4-Pro in this mode to audit whether each package genuinely supports the paper's approach, checking task description and data, evaluator, metadata anchors, and cross-component consistency. After human review, 45 tasks are dropped for systematic defects and 17 receive minor repairs. The benchmark is finalized at 90 task packages.

Reproducibility of the final set. On the finalized 90 tasks, Claude Opus 4.6 reproduces 26 tasks successfully ($g \geq -0.05$) and DeepSeek-V4-Pro reproduces 21 tasks. On the 14 tasks where both succeed, g clusters tightly around zero (median -0.0026, 90% of deviations ≤ 0.034), confirming that the SOTA anchors are well calibrated. Remaining non-successes trace to the uniform resource budget and agent capability rather than package defects (Figure 6).

3.3 Benchmark Statistics

The benchmark comprises 90 tasks and 332 evaluation instances. We characterize NatureBench along two complementary themes below, with full statistics in Appendix E.

Breadth and representativeness. The tasks span six scientific domains and eight ML task types (Figure 3). They concentrate in six journals and skew toward recent papers. Source papers also

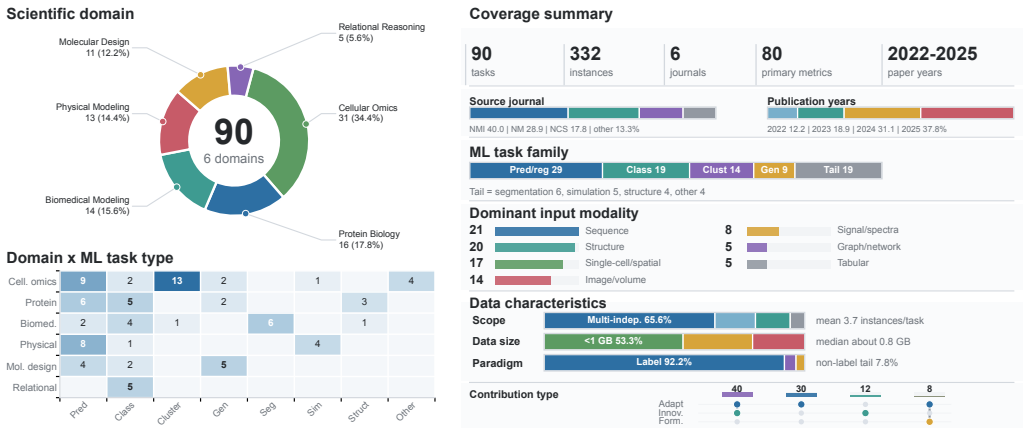

Figure 3: **NatureBench coverage.** Across 90 tasks, NatureBench spans six scientific domains and diverse ML task families while varying substantially in data modality, data characteristics, and source-paper contribution type.

vary in contribution type: most adapt established methods to new scientific settings, a sizable share introduce algorithmic innovations, and a few contribute new problem formulations; a paper can span multiple categories.

Heterogeneous evaluation design. At the *Scope* layer, tasks average 3.7 evaluation instances and use diverse data-partition topologies. At the *Paradigm* layer, tasks fall into three types: *Label*, *Distribution*, and *Oracle*, with *Label* tasks predominating. At the *Metric* layer, tasks use 80 distinct primary metrics. This heterogeneity makes per-task raw scores incomparable, motivating the direction-normalized, scale-free relative-gap metric of §3.4.

3.4 Evaluation Protocol

Each agent solves its task inside an isolated NatureBench container, scored by a standardized evaluation service against the source paper's reported SOTA (Figure 7).

SOTA-normalized relative gap. To compare agents across tasks with heterogeneous metrics, each task is scored and ranked by a single normalized quantity computed on the one primary metric that each instance designates. For instance i, this SOTA-normalized relative gap is

$$g_i \;=\; \mathrm{dir}_i \cdot \frac{m_i - m_i^{\mathrm{sota}}}{|m_i^{\mathrm{sota}}|}, \tag{1}$$

where m_i is the agent's value on that primary metric, m_i^{sota} is the paper-reported SOTA for it, and $\mathrm{dir}_i \in \{+1, -1\}$ encodes the metric direction. $g_i \geq 0$ means the agent matches or surpasses the published result. The task-level score averages g_i across instances, and instances with no valid submission receive $g_i^{\mathrm{fail}} = -1.0$. Other metrics are returned as feedback but do not enter this score.

Agent run and adjudication. The agent receives the task description and data, while the evaluator and ground truth reside in a host-side service that it cannot access. *During* the run, the agent iteratively submits solutions and receives per-instance raw scores, relative gaps, and the running best. The wall clock pauses during scoring. *After* the run, a post-hoc GPT-5.5 judge checks for shortcut behavior and assigns flagged runs a score of `none`. Appendix F provides further details.

4 Experiments

4.1 Experimental Setup

We evaluate 16 frontier coding agents on all 90 tasks under the protocol of §3.4.

Table 2: **Main results on NatureBench,** sorted by overall Surpass-SOTA. Each group reports **S** = Surpass-SOTA ($g > 0.1$) and **M** = Match-SOTA ($g \geq 0$), as percentages of tasks, both overall (All) and per scientific domain. Best/second in the All columns are **bold**/underlined.

	All		Protein		Cellular		Physical		Molec.		Relat.		Biomed.	
Model	S↑	M↑	S↑	M↑	S↑	M↑	S↑	M↑	S↑	M↑	S↑	M↑	S↑	M↑
Claude Opus 5	**23.3**	**57.8**	18.8	62.5	25.8	54.8	30.8	69.2	18.2	36.4	20.0	80.0	21.4	57.1
Claude Opus 4.7	15.6	41.1	12.5	56.2	19.4	45.2	30.8	46.2	9.1	27.3	0.0	60.0	7.1	14.3
Kimi K3	14.4	33.3	6.2	31.2	22.6	38.7	15.4	15.4	9.1	36.4	0.0	60.0	14.3	28.6
GLM-5.2	13.3	35.6	12.5	43.8	22.6	41.9	15.4	15.4	0.0	36.4	0.0	60.0	7.1	21.4
Qwen 3.8 Max	11.1	35.6	6.2	31.2	16.1	38.7	15.4	38.5	9.1	18.2	0.0	80.0	7.1	28.6
Claude Opus 4.6	11.1	28.9	6.2	25.0	19.4	32.3	23.1	30.8	0.0	27.3	0.0	60.0	0.0	14.3
GLM-5.3	10.0	25.6	12.5	31.2	19.4	29.0	0.0	7.7	9.1	27.3	0.0	60.0	0.0	14.3
Gemini 3.5 Flash	10.0	25.6	6.2	31.2	16.1	32.3	23.1	23.1	0.0	18.2	0.0	60.0	0.0	0.0
Kimi K2.6	7.8	27.8	12.5	31.2	12.9	29.0	7.7	7.7	0.0	27.3	0.0	60.0	0.0	28.6
Qwen 3.7 Max	7.8	21.1	12.5	37.5	9.7	22.6	15.4	15.4	0.0	0.0	0.0	60.0	0.0	7.1
MiniMax-M3	6.7	27.8	6.2	37.5	12.9	29.0	7.7	15.4	0.0	36.4	0.0	60.0	0.0	7.1
GLM-5.1	6.7	24.4	6.2	25.0	12.9	35.5	0.0	0.0	0.0	9.1	0.0	60.0	7.1	21.4
GPT-5.4	6.7	24.4	6.2	37.5	9.7	22.6	15.4	23.1	0.0	18.2	0.0	60.0	0.0	7.1
GPT-5.5	4.4	25.6	6.2	31.2	9.7	25.8	0.0	7.7	0.0	18.2	0.0	60.0	0.0	28.6
DeepSeek-V4-Pro	4.4	23.3	6.2	37.5	9.7	29.0	0.0	7.7	0.0	9.1	0.0	60.0	0.0	7.1
MiniMax-M2.7	1.1	12.2	0.0	18.8	3.2	12.9	0.0	7.7	0.0	0.0	0.0	20.0	0.0	14.3

Models. We evaluate sixteen models, each paired with one of three CLI-based agent harnesses. Claude Code (Anthropic, 2025) is paired with thirteen models: Claude Opus 4.6, Claude Opus 4.7, and Claude Opus 5 (Anthropic, 2026a;b;c); Kimi K2.6 and Kimi K3 (Moonshot AI, 2026a;b); MiniMax-M2.7 and MiniMax-M3 (MiniMax, 2026a;b); Qwen 3.7 Max and Qwen 3.8 Max (Qwen Team, 2026a;b); GLM-5.1, GLM-5.2, and GLM-5.3 (Z.ai, 2026a;b;c); and DeepSeek-V4-Pro (DeepSeek, 2026). Codex CLI (OpenAI, 2025) is paired with GPT-5.4 and GPT-5.5 (OpenAI, 2026a;b). Gemini CLI (Google, 2025) is paired with Gemini 3.5 Flash (Google DeepMind, 2026). Each agent is run independently over all 90 tasks.

Unified conditions. All agents disable web search and retain the harness's default reasoning effort. Each task has a 4-hour wall-clock budget, with hardware assigned by its compute requirements in metadata (§2.4): 3 tasks run CPU-only, 70 with lighter GPU requirements each receive 1 NVIDIA RTX 3090 or 4090, and the 17 most compute-intensive each receive 1 NVIDIA A800.

4.2 Main Results

Clear improvements over the published SOTA are rare across all 16 agents. Table 2 reports Surpass-SOTA ($g > 0.1$) and Match-SOTA ($g \geq 0$) rates, both overall and per scientific domain.

Overall performance. Claude Opus 5 leads with 23.3% Surpass-SOTA and 57.8% Match-SOTA, followed by Claude Opus 4.7 (15.6% and 41.1%). Only the strongest matches SOTA on more than half the tasks; MiniMax-M2.7 trails with just 1.1% Surpass-SOTA (Table 2). Attainment varies by scientific domain, with clear improvements more concentrated. We analyze this variation in §5.2.

Completion and validity. Agents submit scorable solutions on nearly all tasks, but the SR–CR gap reveals substantial judge-rejected shortcuts (Table 5). GPT-5.5 shows the largest gap: 98.9% of tasks receive a score, but only 54.4% retain a valid score (40 scored-but-invalid runs). Rejected runs lower its median gap from $\tilde{g}_{\text{valid}} = -0.053$ on judge-accepted tasks to $\tilde{g}_{\text{all}} = -0.462$ across all tasks. GLM-5.1 has the lowest SR (93.3%), with no scorable submission on the remaining tasks.

Score distribution. For most agents, outcomes center modestly below SOTA ($-0.5 \leq g < 0$), while lower-ranked agents show larger shortfalls or invalid/no-score outcomes (Figure 8, Appendix G). Median gaps $\tilde{g}_{\text{all}}$ range from 0.014 for Claude Opus 5 to -0.501 for MiniMax-M2.7 (Table 5). SOTA normalization amplifies large shortfalls on a few tasks into extreme negative gaps, pulling every agent's mean below its median. We therefore prioritize Surpass-SOTA and Match-SOTA, using the median as an auxiliary summary. Appendix H confirms that these extreme values reflect normalization effects rather than faulty tasks.

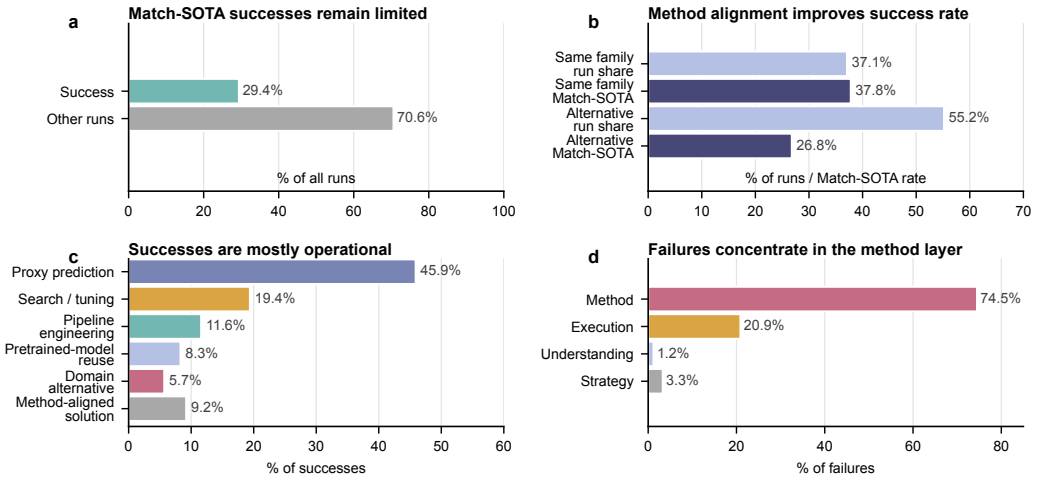

Figure 4: **Solution mechanisms of sixteen evaluated agents across** 1,440 **NatureBench runs.** **a**, Match-SOTA outcomes across all runs. **b**, Match-SOTA rates for runs using the same versus different broad method families as the source paper. **c**, Success-mode distribution among Match-SOTA runs. **d**, Failure-layer distribution among below-SOTA and invalid runs.

5 Analysis

Agents remain far from paper-reported SOTA; we next examine how that gap arises (§5.1) and where it concentrates (§5.2).

5.1 Solution Mechanisms

To understand how agents succeed or fail, we annotate 1,440 runs (90 tasks $\times$ 16 agents), comparing agent and source-paper method families, attributing Match-SOTA runs to success modes, and categorizing below-SOTA or invalid runs into failure layers. Only 29.4% match SOTA (Figure 4a), primarily because this requires methods that fit the task's scientific structure and sufficient execution depth. Appendix J presents three representative trajectories.

Method pathways. Agents systematically reshape scientific tasks into more familiar method families: paper-side methods concentrate in structured representation, statistical modeling, and pretraining or transfer learning, whereas agent-side methods concentrate in supervised predictive modeling (40.6% of runs). Runs using the same broad method family as the source paper match SOTA more often than those using a different family (37.8% versus 26.8%; Figure 4b). Although NatureBench imposes no constraint on method choice, methods closer to the task's original scientific structure tend to be more effective.

Success modes. Success patterns suggest that agents usually match SOTA by reducing the tasks to standard ML pipelines rather than reasoning about their scientific specifics. Engineering-driven categories account for 85.1% of successes: supervised proxy prediction (45.9%), optimization and tuning (19.4%), pipeline engineering (11.6%), and pretrained-model reuse (8.3%). Domain-reasoned alternatives and method-aligned solutions account for only 5.7% and 9.2% (Figure 4c).

Failure modes. Most failed runs produce runnable solutions, but weak methods, invalid shortcuts, or shallow implementation prevent them from matching the paper-reported SOTA; task misunderstanding and malformed outputs are less common. Among the 70.6% of runs that fall below Match-SOTA or lack a valid score, method-layer failures dominate at 74.5%, primarily wrong method choice (29.8%) and judge-identified shortcuts (25.5%), followed by execution-layer failures at 20.9%, largely due to insufficient budget or time (17.6%). Strategy-layer and understanding-layer failures account for only 3.3% and 1.2%, respectively (Figure 4d). Appendix I examines whether larger time budgets reduce failure rates.

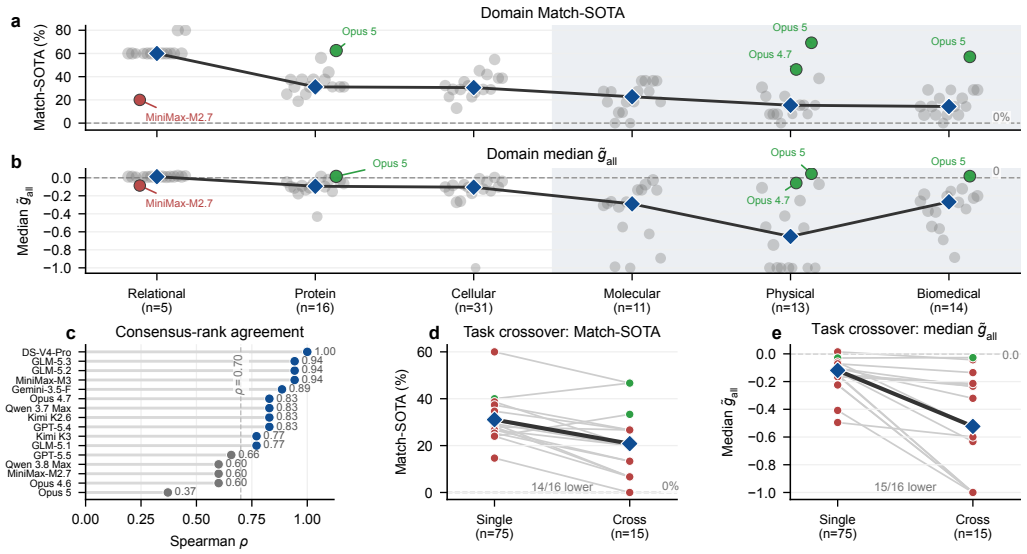

Figure 5: **Agent performance by scientific domain and disciplinary scope. a,b,** Match-SOTA rate and median $\tilde{g}_{\text{all}}$ for 16 agents across six domains. Grey circles: agents; blue diamonds: domain medians. Green/red circles indicate above/below-median Match-SOTA rates by at least 30 percentage points, with same-direction $\tilde{g}_{\text{all}}$ deviations. **c,** Spearman ρ of agent domain rankings against the Match-SOTA consensus. **d,e,** The same metrics for 75 single- versus 15 cross-discipline tasks. Diamonds: across-agent mean (d) and median (e); green/red pairs: increases/decreases.

5.2 Domain and Interdisciplinary Variation

Most agents share a similar difficulty ordering across scientific domains, while interdisciplinary tasks show wider gaps to paper-reported SOTA (Figure 5).

Scientific domain. The consensus Match-SOTA rate (the median across agents) separates an easier tier of Relational Reasoning (60.0%), Protein Biology (31.2%), and Cellular Omics (30.6%) from a harder tier of Molecular Design (22.7%), Physical Modeling (15.4%), and Biomedical Modeling (14.3%). The consensus $\tilde{g}_{\text{all}}$ corroborates this split, remaining above -0.11 for the easier tier and below -0.26 for the harder tier. Eleven of sixteen agents broadly follow this ordering (Spearman $\rho \geq 0.77$); Claude Opus 5 is the clearest exception, performing particularly well in Physical and Biomedical Modeling.

Interdisciplinary tasks. The 90 tasks comprise 15 cross-discipline and 75 single-discipline tasks. The pooled median $\tilde{g}_{\text{all}}$ falls from -0.12 on single-discipline tasks to -0.52 on cross-discipline tasks, with 15 of 16 agents moving in this direction. The Match-SOTA rate likewise drops from 31.1% to 20.8%, with 14 of 16 agents lower. These consistently wider agent–SOTA gaps suggest that integrating knowledge across domains remains a distinct challenge for most current agents.

6 Conclusion

We introduced NatureGym, an automated pipeline that constructs scientific task environments from Nature-family papers, and NatureBench, which uses these environments to evaluate *discovery* beyond reproduction. Across 16 frontier agents, clear improvements over the published SOTA remain rare; even the best matches it on only 57.8% of tasks. The dominant success pathway is methodological translation, where agents convert scientific tasks into familiar supervised-prediction problems rather than scientific invention. Failures are dominated by wrong method choice and judge-identified shortcuts, followed by insufficient compute budget, not task misunderstanding. We release NatureBench, NatureGym, and a public leaderboard with maintainer-side reproduction, with the long-term aim of turning the same substrate into training data for future scientific-discovery agents.

AI USE STATEMENT

We used LLM agents with skills in NatureGym for paper filtering, dataset preparation, and task-package construction. We also used LLMs to assist with trajectory analysis and interpretation, and GPT-5.5 for post-hoc validity adjudication, as described in the paper. Generative AI additionally assisted with literature search, code development, figure preparation, and manuscript revision. The authors take responsibility for the final text, analysis, and artifacts, including those produced with AI assistance.

ETHICS STATEMENT

NatureBench draws on existing scientific publications and publicly accessible datasets. Our releases distinguish original benchmark contributions from third-party materials, which we use and distribute in accordance with their original licenses and terms of use. We also distinguish each benchmark task from the full scientific contribution of its source paper. Tasks retain the paper's core quantitative problem but exclude wet-lab experiments and other steps requiring physical interaction, and may cover only a subset of its evaluation instances and metrics due to requirements such as data availability and feasibility of automated evaluation. Accordingly, Match-SOTA and Surpass-SOTA refer to performance within the retained task scope and should not be interpreted as matching or surpassing the source paper's entire scientific contribution (Appendix H).

REPRODUCIBILITY STATEMENT

We support reproducibility by releasing the NatureGym construction pipeline, the NatureBench task packages, and the evaluation framework. Each task package includes the task description, data, evaluator, environment specification, and paper-reported SOTA scores. To complement the procedures outlined in the main text, we provide detailed descriptions of task construction and verification in Appendix B and of scoring and execution in Appendix F. Appendix D further documents how task packages are calibrated before evaluation, while §4.1 specifies the model–harness configurations and time and hardware budgets. We also report token usage and estimated costs in Appendix K to help researchers assess the resources needed to repeat the evaluations.

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.

Anthropic. Claude code: An agentic coding tool. https://github.com/anthropics/claude-code, 2025.

Anthropic. System card: Claude opus 4.6. https://www.anthropic.com/claude-opus-4-6-system-card, 2026a.

Anthropic. System card: Claude opus 4.7. https://www.anthropic.com/claude-opus-4-7-system-card, 2026b.

Anthropic. Introducing claude opus 5. https://www.anthropic.com/news/claude-opus-5, 2026c.

Anthropic. Claude api pricing. https://platform.claude.com/docs/en/about-claude/pricing, 2026d.

Akari Asai, Jacqueline He, Rulin Shao, Weijia Shi, Amanpreet Singh, Joseph Chee Chang, Kyle Lo, Luca Soldaini, Sergey Feldman, Mike D'Arcy, David Wadden, Matt Latzke, Minyang Tian, Pan Ji, Shengyan Liu, Hao Tong, Bohao Wu, Yanyu Xiong, Luke Zettlemoyer, Graham Neubig, Dan Weld, Doug Downey, Wen tau Yih, Pang Wei Koh, and Hannaneh Hajishirzi. OpenScholar: Synthesizing scientific literature with retrieval-augmented LMs, 2024. URL https://arxiv.org/abs/2411.14199.

Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N Kinch, R Dustin Schaeffer, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021.

Peizhen Bai, Filip Miljković, Xianyuan Liu, Leonardo De Maria, Rebecca Croasdale-Wood, Owen Rackham, and Haiping Lu. Mask-prior-guided denoising diffusion improves inverse protein folding. *Nature Machine Intelligence*, 7(6):876–888, 2025.

Nathaniel R Bennett, Joseph L Watson, Robert J Ragotte, Andrew J Borst, DéJenaé L See, Connor Weidle, Riti Biswas, Yutong Yu, Ellen L Shrock, Russell Ault, et al. Atomically accurate de novo design of antibodies with rfdiffusion. *Nature*, 649(8095):183–193, 2026.

Cristian Bodnar, Wessel P Bruinsma, Ana Lucic, Megan Stanley, Anna Allen, Johannes Brandstetter, Patrick Garvan, Maik Riechert, Jonathan A Weyn, Haiyu Dong, et al. A foundation model for the earth system. *Nature*, 641(8065):1180–1187, 2025.

Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.

Garyk Brixi, Matthew G Durrant, Jerome Ku, Mohsen Naghipourfar, Michael Poli, Gwanggyu Sun, Greg Brockman, Daniel Chang, Alison Fanton, Gabriel A Gonzalez, et al. Genome modelling and design across all domains of life with evo 2. *Nature*, 652(8112):1349–1361, 2026.

Mert Cemri, Shubham Agrawal, Akshat Gupta, Shu Liu, Audrey Cheng, Qiuyang Mang, Ashwin Naren, Lutfi Eren Erdogan, Koushik Sen, Matei Zaharia, Alex Dimakis, and Ion Stoica. AdaEvolve: Adaptive LLM driven zeroth-order optimization, 2026. URL https://arxiv.org/abs/2602.20133.

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

Shuan Chen and Yousung Jung. A generalized-template-based graph neural network for accurate organic reactivity prediction. *Nature Machine Intelligence*, 4(9):772–780, 2022. doi: 10.1038/s42256-022-00526-z.

Jun Cheng, Guido Novati, Joshua Pan, Clare Bycroft, Akvilė Žemgulytė, Taylor Applebaum, Alexander Pritzel, Lai Hong Wong, Michal Zielinski, Tobias Sargeant, et al. Accurate proteome-wide missense variant effect prediction with alphamissense. *Science*, 381(6664):eadg7492, 2023.

Yizhe Chi, Deyao Hong, Dapeng Jiang, Tianwei Luo, Kaisen Yang, Boshi Zhang, Zhe Cao, Xiaoyan Fan, Bingxiang He, Han Hao, Weiyang Jin, Dianqiao Lei, Qingle Liu, Houde Qian, Bowen Wang, Situ Wang, Youjie Zheng, Yifan Zhou, Calvin Xiao, Eren Cai, and Qinhuai Na. Frontier-Eng: Benchmarking self-evolving agents on real-world engineering tasks with generative optimization, 2026. URL https://arxiv.org/abs/2604.12290.

Haotian Cui, Chloe Wang, Hassaan Maan, Kuan Pang, Fengning Luo, Nan Duan, and Bo Wang. scgpt: toward building a foundation model for single-cell multi-omics using generative ai. *Nature methods*, 21(8):1470–1480, 2024.

Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza-Revilla, Nicolas Lopez Carranza, Adam Henryk Grzywaczewski, Francesco Oteri, Christian Dallago, Evan Trop, Bernardo P. de Almeida, Hassan Sirelkhatim, Guillaume Richard, Marcin Skwark, Karim Beguir, Marie Lopez, and Thomas Pierrot. Nucleotide transformer: building and evaluating robust foundation models for human genomics. *Nature Methods*, 22:287–297, 2025. doi: 10.1038/s41592-024-02523-z.

DeepSeek. Deepseek v4 preview release. https://api-docs.deepseek.com/news/news260424, 2026.

Darius A. Faroughy, Sofia Palacios Schweitzer, Ian Pang, Siddharth Mishra-Sharma, and David Shih. Collider-bench: Benchmarking AI agents with particle physics analysis reproduction, 2026. URL https://arxiv.org/abs/2605.13950.

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

Xian Gao, Jiacheng Ruan, Zongyun Zhang, Jingsheng Gao, Ting Liu, and Yuzhuo Fu. MMReview: A multidisciplinary and multimodal benchmark for LLM-based peer review automation, 2025. URL https://arxiv.org/abs/2508.14146.

Ali Essam Ghareeb, Benjamin Chang, Ludovico Mitchener, et al. A multi-agent system for automating scientific discovery. *Nature*, 2026. doi: 10.1038/s41586-026-10652-y. URL https://www.nature.com/articles/s41586-026-10652-y.

Google. Gemini cli: An open-source ai agent. https://github.com/google-gemini/gemini-cli, 2025.

Google DeepMind. Gemini 3.5 flash model card. https://deepmind.google/models/model-cards/gemini-3-5-flash/, 2026.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Petar Sirkovic, Artiom Myaskovsky, Grzegorz Glowaty, Felix Weissenberger, Alessio Orlandi, Dan Popovici, et al. Accelerating scientific discovery with Co-Scientist. *Nature*, 2026. doi: 10.1038/s41586-026-10644-y. URL https://www.nature.com/articles/s41586-026-10644-y.

Qianyue Hao, Fengli Xu, Yong Li, and James Evans. Artificial intelligence tools expand scientists' impact but contract science's focus. *Nature*, pp. 1–7, 2026.

Ramin Hasani, Mathias Lechner, Alexander Amini, Lucas Liebenwein, Aaron Ray, Max Tschaikowski, Gerald Teschl, and Daniela Rus. Closed-form continuous-time neural networks. *Nature Machine Intelligence*, 4(11):992–1003, 2022.

Chuxuan Hu, Liyun Zhang, Yeji Lim, Aum Wadhwani, Austin Peters, and Daniel Kang. REPRO-bench: Can agentic AI systems assess the reproducibility of social science research?, 2025. URL https://arxiv.org/abs/2507.18901.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating language agents on machine learning experimentation, 2023. URL https://arxiv.org/abs/2310.03302.

Ziyang Huang, Yi Cao, Ali K. Shargh, Jing Luo, Ruidong Mei, Mohd Zaki, Zhan Liu, Wyatt Bunstine, William Jurayj, Somdatta Goswami, Tyrel McQueen, Michael Shields, Jaafar El-Awady, Paulette Clancy, Benjamin Van Durme, Nicholas Andrews, William Walden, and Daniel Khashabi. Can coding agents reproduce findings in computational materials science?, 2026. URL https://arxiv.org/abs/2605.00803.

Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain Masoom, et al. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, pp. 1–3, 2025.

Ilia Igashov, Hannes Stärk, Clément Vignac, Arne Schneuing, Victor Garcia Satorras, Pascal Frossard, Max Welling, Michael Bronstein, and Bruno Correia. Equivariant 3d-conditional diffusion model for molecular linker design. *Nature Machine Intelligence*, 6(4):417–427, 2024.

Yuki Imajuku, Kohki Horie, Yoichi Iwata, Kensho Aoki, Naohiro Takahashi, and Takuya Akiba. ALE-bench: A benchmark for long-horizon objective-driven algorithm engineering. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2025. URL https://openreview.net/forum?id=JCjGvbsOmQ.

Peter Jansen, Marc-Alexandre Cote, Tushar Khot, Erin Bransom, Bhavana Dalvi Mishra, Bodhisattwa Prasad Majumder, Oyvind Tafjord, and Peter Clark. DISCOVERYWORLD: A virtual environment for developing and evaluating automated scientific discovery agents, 2024. URL https://arxiv.org/abs/2406.06769.

Jiachen Jiang, Tianyu Ding, and Zhihui Zhu. DeltaEvolve: Accelerating scientific discovery through momentum-driven evolution, 2026. URL https://arxiv.org/abs/2602.02919.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

Andrej Karpathy. autoresearch. https://github.com/karpathy/autoresearch, 2026.

Gyeongwon James Kim, Alex Wilf, Louis-Philippe Morency, and Daniel Fried. From reproduction to replication: Evaluating research agents with progressive code masking, 2025. URL https://arxiv.org/abs/2506.19724.

Vitalii Kleshchevnikov, Artem Shmatko, Emma Dann, Alexander Aivazidis, Hamish W King, Tong Li, Rasa Elmentaite, Artem Lomakin, Veronika Kedlian, Adam Gayoso, et al. Cell2location maps fine-grained cell types in spatial transcriptomics. *Nature biotechnology*, 40(5):661–671, 2022.

Jakub Lala, Odhran O'Donoghue, Aleksandar Shtedritski, Sam Cox, Samuel G. Rodriques, and Andrew D. White. PaperQA: Retrieval-augmented generative agent for scientific research, 2023. URL https://arxiv.org/abs/2312.07559.

Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

Jon M. Laurent, Joseph D. Janizek, Michael Ruzo, Michaela M. Hinks, Michael J. Hammerling, Siddharth Narayanan, Manvitha Ponnapati, Andrew D. White, and Samuel G. Rodriques. LAB-bench: Measuring capabilities of language models for biology research, 2024. URL https://arxiv.org/abs/2407.10362.

Yu Li, Chenyang Shao, Xinyang Liu, Ruotong Zhao, Peijie Liu, Hongyuan Su, Zhibin Chen, Qinglong Yang, Anjie Xu, Yi Fang, Qingbin Zeng, Tianxing Li, Jingbo Xu, Fengli Xu, Yong Li, and Tie-Yan Liu. AutoSOTA: An end-to-end automated research system for state-of-the-art AI model discovery, 2026. URL https://arxiv.org/abs/2604.05550.

Weixin Liang, Yuhui Zhang, Hancheng Cao, Binglu Wang, Daisy Ding, Xinyu Yang, Kailas Vodrahalli, Siyu He, Daniel Smith, Yian Yin, Daniel McFarland, and James Zou. Can large language models provide useful feedback on research papers? a large-scale empirical analysis, 2023. URL https://arxiv.org/abs/2310.01783.

Minhua Lin, Hanqing Lu, Zhan Shi, Bing He, Rui Mao, Zhiwei Zhang, Zongyu Wu, Xianfeng Tang, Hui Liu, Zhenwei Dai, Xiang Zhang, Suhang Wang, Benoit Dumoulin, and Jian Pei. Position: Agentic evolution is the path to evolving LLMs, 2026. URL https://arxiv.org/abs/2602.00359.

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.

Gang Liu, Yihan Zhu, et al. Scientific algorithm discovery by augmenting AlphaEvolve with deep research, 2025. URL https://arxiv.org/abs/2510.06056.

Ryan Liu and Nihar B. Shah. ReviewerGPT? an exploratory study on using large language models for paper reviewing, 2023. URL https://arxiv.org/abs/2306.00622.

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, 2024. URL https://arxiv.org/abs/2408.06292.

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.

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

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

Yougang Lyu, Xi Zhang, Xinhao Yi, Yuyue Zhao, Shuyu Guo, Wenxiang Hu, Jan Piotrowski, Jakub Kaliski, Jacopo Urbani, Zaiqiao Meng, Lun Zhou, and Xiaohui Yan. EvoScientist: Towards multi-agent evolving AI scientists for end-to-end scientific discovery, 2026b. URL https://arxiv.org/abs/2603.08127.

Qiuyang Mang, Wenhao Chai, Zhifei Li, Huanzhi Mao, Shang Zhou, Alexander Du, Hanchen Li, Shu Liu, Edwin Chen, Yichuan Wang, Xieting Chu, Zerui Cheng, Yuan Xu, Tian Xia, Zirui Wang, Tianneng Shi, Jianzhu Yao, Yilong Zhao, Qizheng Zhang, Charlie Ruan, Zeyu Shen, Kaiyuan Liu, Runyuan He, Dong Xing, Zerui Li, Zirong Zeng, Yige Jiang, Lufeng Cheng, Ziyi Zhao, Youran Sun, Wesley Zheng, Meiyuwang Zhang, Ruyi Ji, Xuechang Tu, Zihan Zheng, Zexing Chen, Kangyang Zhou, Zhaozi Wang, Jingbang Chen, Aleksandra Korolova, Peter Henderson, Pramod Viswanath, Vijay Ganesh, Saining Xie, Zhuang Liu, Dawn Song, Sewon Min, Ion Stoica, Joseph E. Gonzalez, Jingbo Shang, and Alvin Cheung. FrontierCS: Evolving challenges for evolving intelligence, 2025. URL https://arxiv.org/abs/2512.15699.

Amil Merchant, Simon Batzner, Samuel S Schoenholz, Muratahan Aykol, Gowoon Cheon, and Ekin Dogus Cubuk. Scaling deep learning for materials discovery. *Nature*, 624(7990):80–85, 2023.

Jishuai Miao, Jinzhao Li, Jingxue Xin, Jiajuan Tu, Muyang Ge, Ji Qi, Xiaocheng Zhou, Ying Zhu, Can Yang, and Zhixiang Lin. Multigate: integrative analysis and regulatory inference in spatial multi-omics data via graph representation learning. *Nature Communications*, 16(1):9403, 2025.

MiniMax. Minimax m2.7: Early echoes of self-evolution. https://www.minimax.io/news/minimax-m27-en, 2026a.

MiniMax. Minimax m3: Coding & agentic frontier. https://www.minimax.io/models/text/m3, 2026b.

Moonshot AI. Kimi k2.6. https://www.kimi.com/ai-models/kimi-k2-6, 2026a.

Moonshot AI. Kimi k3: Open frontier intelligence. https://www.kimi.com/blog/kimi-k3, 2026b.

Deepak Nathani, Lovish Madaan, Nicholas Roberts, Nikolay Bashlykov, Ajay Menon, Vincent Moens, Amar Budhiraja, Despoina Magka, Vladislav Vorotilov, Gaurav Chaurasia, Dieuwke Hupkes, Ricardo Silveira Cabral, Tatiana Shavrina, Jakob Foerster, Yoram Bachrach, William Yang Wang, and Roberta Raileanu. MLGym: A new framework and benchmark for advancing AI research agents, 2025. URL https://arxiv.org/abs/2502.14499.

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

OpenAI. Codex cli: Lightweight coding agent that runs in your terminal. https://github.com/openai/codex, 2025.

OpenAI. Gpt-5.4 thinking system card. https://openai.com/index/gpt-5-4-thinking-system-card/, 2026a.

OpenAI. Gpt-5.5 system card. https://openai.com/index/gpt-5-5-system-card/, 2026b.

OpenAI. What are tokens and how to count them? https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them, 2026c.

Jens Oppliger, M Michael Denner, Julia Küspert, Ruggero Frison, Qisi Wang, Alexander Morawietz, Oleh Ivashko, Ann-Christin Dippel, Martin von Zimmermann, Izabela Biało, et al. Weak signal extraction enabled by deep neural network denoising of diffraction data. *Nature Machine Intelligence*, 6(2):180–186, 2024.

Jinghua Piao, Jiazhen Liu, Fang Zhang, Jun Su, and Yong Li. Human–ai adaptive dynamics drives the emergence of information cocoons. *Nature Machine Intelligence*, 5(11):1214–1224, 2023.

Jesús Pineda, Sergi Masó-Orriols, Montse Masoliver, Joan Bertran, Mattias Goksör, Giovanni Volpe, and Carlo Manzo. Enhanced spatial clustering of single-molecule localizations with graph neural networks. *Nature Communications*, 16(1):9693, 2025.

Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Tom R Andersson, Andrew El-Kadi, Dominic Masters, Timo Ewalds, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, et al. Probabilistic weather forecasting with machine learning. *Nature*, 637(8044):84–90, 2025.

Rushi Qiang, Yuchen Zhuang, Yinghao Li, Dingu Sagar VK, Rongzhi Zhang, Changhao Li, Ian Wong, Sherry Yang, Percy Liang, Chao Zhang, et al. Mle-dojo: Interactive environments for empowering llm agents in machine learning engineering. *Advances in Neural Information Processing Systems*, 38, 2026.

Qwen Team. Qwen3.7: The agent frontier. https://qwen.ai/blog?id=qwen3.7, 2026a.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork. https://qwen.ai/blog?id=qwen3.8, August 2026b.

Ben Rank, Hardik Bhatnagar, Ameya Prabhu, Shira Eisenberg, Karina Nguyen, Matthias Bethge, and Maksym Andriushchenko. PostTrainBench: Can LLM agents automate LLM post-training?, 2026. URL https://arxiv.org/abs/2603.08640.

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

Zachary S. Siegel, Sayash Kapoor, Nitya Nagdir, Benedikt Stroebl, and Arvind Narayanan. CORE-bench: Fostering the credibility of published research through a computational reproducibility agent benchmark, 2024. URL https://arxiv.org/abs/2409.11363.

Michael D. Skarlinski, Sam Cox, Jon M. Laurent, James D. Braza, Michaela Hinks, Michael J. Hammerling, Manvitha Ponnapati, Samuel G. Rodriques, and Andrew D. White. Language agents achieve superhuman synthesis of scientific knowledge, 2024. URL https://arxiv.org/abs/2409.13740.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. PaperBench: Evaluating AI's ability to replicate AI research, 2025. URL https://arxiv.org/abs/2504.01848.

Xiaorui Su, Pengwei Hu, Dongxu Li, Bowei Zhao, Zhaomeng Niu, Thomas Herget, Philip S. Yu, and Lun Hu. Interpretable identification of cancer genes across biological networks via transformer-powered graph representation learning. *Nature Biomedical Engineering*, 9(3):371–389, 2025. doi: 10.1038/s41551-024-01312-5.

Christina V Theodoris, Ling Xiao, Anant Chopra, Mark D Chaffin, Zeina R Al Sayed, Matthew C Hill, Helene Mantineo, Elizabeth M Brydon, Zexian Zeng, X Shirley Liu, et al. Transfer learning enables predictions in network biology. *Nature*, 618(7965):616–624, 2023.

Yiping Wang, Shao-Rong Su, Zhiyuan Zeng, Eva Xu, Liliang Ren, Xinyu Yang, Zeyi Huang, Xuehai He, Luyao Ma, Baolin Peng, Hao Cheng, Pengcheng He, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. ThetaEvolve: Test-time learning on open problems, 2025. URL https://arxiv.org/abs/2511.23473.

Zhen Wang, Fan Bai, Zhongyan Luo, Jinyan Su, Kaiser Sun, Xinle Yu, Jieyuan Liu, Kun Zhou, Claire Cardie, Mark Dredze, Eric P. Xing, and Zhiting Hu. FIRE-bench: Evaluating agents on the rediscovery of scientific insights, 2026. URL https://arxiv.org/abs/2602.02905.

Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

Jun Wen, Sihang Zeng, Clara-Lea Bonzel, Shilpa Nadimpalli Kobren, Jiangchuan Du, Yi Chai, Hao Wang, Meng Zhu, Siwei Chen, Fangwei Leng, et al. Phenotypic prediction of missense variants via deep contrastive learning. *Nature Biomedical Engineering*, pp. 1–16, 2026.

Yixuan Weng, Minjun Zhu, Qiujie Xie, Qiyao Sun, Zhen Lin, Sifan Liu, and Yue Zhang. DeepScientist: Advancing frontier-pushing scientific findings progressively, 2025. URL https://arxiv.org/abs/2509.26603.

Jeremy Wohlwend, Gabriele Corso, Saro Passaro, Noah Getz, Mateo Reveiz, Ken Leidal, Wojtek Swiderski, Liam Atkinson, Tally Portnoi, Itamar Chinn, et al. Boltz-1 democratizing biomolecular interaction modeling. *BioRxiv*, pp. 2024–11, 2025.

Zhangchen Xu, Junda Chen, Yue Huang, Dongfu Jiang, Jiefeng Chen, Hang Hua, Zijian Wu, Zheyuan Liu, Zexue He, Lichi Li, et al. Autolab: Can frontier models solve long-horizon auto research and engineering tasks? *arXiv preprint arXiv:2606.05080*, 2026.

Shuo Yan, Ruochen Li, Ziming Luo, Zimu Wang, Daoyang Li, Liqiang Jing, Kaiyu He, Peilin Wu, George Michalopoulos, Yue Zhang, Ziyang Zhang, Mian Zhang, Zhiyu Chen, and Xinya Du. LMR-BENCH: Evaluating LLM agent's ability on reproducing language modeling research, 2025. URL https://arxiv.org/abs/2506.17335.

Christine Ye, Sihan Yuan, Suchetha Cooray, Steven Dillmann, Ian L. V. Roque, Dalya Baron, Philipp Frank, Sergio Martin-Alvarez, Nolan Koblischke, Frank J Qu, Diyi Yang, Risa Wechsler, and Ioana Ciuca. ReplicationBench: Can AI agents replicate astrophysics research papers?, 2025. URL https://arxiv.org/abs/2510.24591.

Haotian Ye, Haowei Lin, Jingyi Tang, Yizhen Luo, Caiyin Yang, Chang Su, Rahul Thapa, Rui Yang, Ruihua Liu, Zeyu Li, Chong Gao, Dachao Ding, Guangrong He, Miaolei Zhang, Lina Sun, Wenyang Wang, Yuchen Zhong, Zhuohao Shen, Di He, Jianzhu Ma, Stefano Ermon, Tongyang Li, Xiaowen Chu, James Zou, and Yuzhi Xu. Evaluation-driven scaling for scientific discovery, 2026. URL https://arxiv.org/abs/2604.19341.

Jehyeok Yeon, Ben Rank, and Maksym Andriushchenko. InferenceBench: Benchmarking open-ended inference optimization by AI agents, 2026. URL https://inferencebench.ai/.

Mert Yuksekgonul, Daniel Koceja, Xinhao Li, Federico Bianchi, Jed McCaleb, Xiaolong Wang, Jan Kautz, Yejin Choi, James Zou, Carlos Guestrin, and Yu Sun. Learning to discover at test time, 2026. URL https://arxiv.org/abs/2601.16175.

Z.ai. Glm-5.1: Towards long-horizon tasks. https://z.ai/blog/glm-5.1, 2026a.

Z.ai. Glm-5.2: Built for long-horizon tasks. https://z.ai/blog/glm-5.2, 2026b.

Z.ai. Glm-5.3: Frontier coding with emergent cyber capabilities. https://z.ai/blog/glm-5.3, 2026c.

Claudio Zeni, Robert Pinsler, Daniel Zügner, Andrew Fowler, Matthew Horton, Xiang Fu, Zilong Wang, Aliaksandra Shysheya, Jonathan Crabbé, Shoko Ueda, et al. A generative model for inorganic materials design. *Nature*, 639(8055):624–632, 2025.

Zhengxin Zhang, Ning Wang, Sainyam Galhotra, and Claire Cardie. How far are we from true auto-research?, 2026. URL https://arxiv.org/abs/2605.19156.

Bowen Zhou, Ning Ding, Lei Bai, and Hao Zhou. Advancing ai for science: From the revolution of tools to the tools for revolution. *AI Open*, 2025.

Xinyu Zhu, Yuzhu Cai, Zexi Liu, Cheng Wang, Fengyang Li, Wenkai Jin, Wanxu Liu, Zehao Bing, Bingyang Zheng, Jingyi Chai, et al. Evomaster: A foundational evolving agent framework for agentic science at scale. *arXiv preprint arXiv:2604.17406*, 2026.

A Related Work

A.1 AI for Science

The first wave: AI as an accelerator within human-defined research programs. AI for Science has produced strong vertical results across many disciplines. In structural biology, AlphaFold, RoseTTAFold, ESMFold, AlphaFold 3, and Boltz-1 expand atomic-level prediction from single chains to biomolecular complexes (Jumper et al., 2021; Baek et al., 2021; Lin et al., 2023; Abramson et al., 2024; Wohlwend et al., 2025), while RFdiffusion and its antibody extension close the loop with experimentally validated *de novo* design (Watson et al., 2023; Bennett et al., 2026). In genomics, AlphaMissense and PheMART model variant pathogenicity and phenotype space (Cheng et al., 2023; Wen et al., 2026). Geneformer, scGPT, and Evo 2 pretrain foundation models over transcriptomes or DNA (Theodoris et al., 2023; Cui et al., 2024; Brixi et al., 2026). Cell2location resolves cell types in spatial transcriptomics (Kleshchevnikov et al., 2022). In materials, chemistry, mathematics, and Earth systems, GNoME and MatterGen discover or inverse-design materials (Merchant et al., 2023; Zeni et al., 2025), Coscientist automates chemical experimentation (Boiko et al., 2023), AlphaTensor and AlphaProof extend search-based reasoning to algorithms and formal mathematics (Fawzi et al., 2022; Hubert et al., 2025), and GraphCast, GenCast, and Aurora advance global weather and Earth-system prediction (Lam et al., 2023; Price et al., 2025; Bodnar et al., 2025).

A structural limitation of the *research-plus-AI* paradigm. Powerful as these systems are, they mostly share the same methodological form: humans specify the research programme, curate the data, and fix the success criterion, while AI acts as a more capable instrument inside that programme. This makes many advances a *revolution of tools* rather than a *tool of revolution* (Zhou et al., 2025). Large-scale publication evidence further suggests that AI-augmented science can raise individual output and impact while narrowing the collective topic frontier toward data-rich subfields (Hao et al., 2026). Thus, existing AI-for-Science systems can accelerate progress along established axes, but they do not by themselves establish cross-disciplinary, paradigm-shifting problem solving.

From AI-assisted research to AI-native problem solving. The natural next step is to evaluate AI as the primary problem solver: given a scientific task, the system must choose methods, run experiments, and be judged by the final scientific outcome. General-purpose scientific agents such as The AI Scientist, the AI co-scientist, DeepScientist, and AutoSOTA move in this direction (Lu et al., 2026; Gottweis et al., 2026; Weng et al., 2025; Li et al., 2026), but they are usually demonstrated on self-selected topics or within limited domains, leaving open whether AI-native problem solving generalizes across science as a whole.

Cross-disciplinary evaluation as a test of breaking the information cocoon. Contemporary scientists face an increasingly restrictive information cocoon: specialized training, literature growth, and field-specific tooling make it difficult to integrate methods, data, and concepts across disciplines (Hao et al., 2026; Zhou et al., 2025; Piao et al., 2023). This is where an AI-native solver should have a distinctive advantage, because the same agent can combine biological representation learning, chemical search, physical simulation, and statistical modeling within one system. NatureBench therefore tests the missing horizontal capability: whether contemporary coding agents can solve 90 Nature-family tasks across six scientific task domains, using each paper's reported SOTA as a unified discovery scoring anchor and evaluating whether agents can move beyond field-specific *research-plus-AI* toward cross-disciplinary scientific problem solving.

A.2 Paper-based Benchmarks

The paper-based benchmark literature asks whether agents can read, evaluate, and operationalize scientific papers as the core artifact. One line targets paper understanding: PaperQA, PaperQA2, and OpenScholar evaluate retrieval-augmented, citation-backed answers or literature syntheses (Lala et al., 2023; Skarlinski et al., 2024; Asai et al., 2024). LAB-Bench extends this to biology papers with supplementary materials, figures, tables, and protocols (Laurent et al., 2024). ReviewerGPT, large-scale LLM-feedback studies, and MMReview test peer-review-style critique over text-only, multidisciplinary, or multimodal manuscripts (Liu & Shah, 2023; Liang et al., 2023; Gao et al., 2025).

A second line turns papers into executable work: PaperBench asks agents to reconstruct ICML papers from scratch under author-informed rubrics (Starace et al., 2025), while AutoExperiment and LMR-Bench use progressive code masking or language-modeling research specifications to test recovery of reported experiments (Kim et al., 2025; Yan et al., 2025). Reproducibility benchmarks broaden this beyond ML: CORE-Bench, REPRO-Bench, and ReplicationBench cover reproduction, assessment, or replication across computer science, social science, medicine, and astrophysics (Siegel et al., 2024; Hu et al., 2025; Ye et al., 2025). AutoMat and Collider-Bench add materials-science and Large Hadron Collider toolchains (Huang et al., 2026; Faroughy et al., 2026). FIRE-Bench asks agents to rediscover established insights from high-level questions extracted from ML papers (Wang et al., 2026). These benchmarks ground evaluation in papers, but their target is reading, review, reproduction, replication, reproducibility assessment, or rediscovery of known findings. NatureBench keeps paper grounding while shifting the target to independently solving the same scientific problem, using the source paper's reported SOTA as the scoring anchor to match or surpass.

A.3 AI-train-AI and Autonomous Optimization

Recent AI-train-AI, autonomous-optimization, and auto-research work can be organized by how it models the agent's task. Benchmark suites such as MLAgentBench, MLE-bench, MLGym, MLE-Dojo, MLS-Bench, AIRS-Bench, PostTrainBench, InferenceBench, and AutoLab evaluate agents over collections of ML experimentation, model-building, post-training, inference-optimization, or long-horizon closed-loop optimization tasks (Huang et al., 2023; Chan et al., 2025; Nathani et al., 2025; Qiang et al., 2026; Lyu et al., 2026a; Lupidi et al., 2026; Rank et al., 2026; Yeon et al., 2026; Xu et al., 2026). FrontierCS, ALE-Bench, and Frontier-Eng extend this suite-style evaluation to algorithm engineering and real-world engineering optimization (Mang et al., 2025; Imajuku et al., 2025; Chi et al., 2026). A second line studies few-task, verifier-driven discovery, where agents repeatedly propose, execute, and evaluate programs, algorithms, or scientific candidates on specialized high-value objectives (Fawzi et al., 2022; Romera-Paredes et al., 2024; Novikov et al., 2025; Wang et al., 2025; Yuksekgonul et al., 2026; Ye et al., 2026; Cemri et al., 2026; Liu et al., 2026; Jiang et al., 2026; Lin et al., 2026; Liu et al., 2025). A third line frames the task as end-to-end research automation, including simulated scientific environments, autonomous paper-generation workflows, multi-agent hypothesis generation, lab-in-the-loop discovery, SOTA model discovery, and reviewer-style evaluation of generated research (Jansen et al., 2024; Lu et al., 2024; Gottweis et al., 2026; Ghareeb et al., 2026; Li et al., 2026; Weng et al., 2025; Zhang et al., 2026; Lyu et al., 2026b; Zhu et al., 2026). These task models leave the key intersection underexplored: large-scale benchmark suites grounded in paper-level scientific research and evaluated against the paper's reported SOTA on its core scientific metric. NatureBench fills this gap with 90 Nature-family tasks that combine benchmark-suite scale, paper-sourced science, and SOTA-referenced evaluation across six scientific task domains.

B NatureGym Details

This appendix provides construction and verification details for NatureGym, complementing the main-text overview.

B.1 Paper Filtering

This subsection details the paper filtering procedures and criteria outlined in §2.2.

Preprocessing. Each paper is converted into three structured components that the subsequent filtering stages consume. After retaining only research articles and dropping non-research content (e.g., news, editorials, corrections, reviews), we produce from each article: (i) markdown text preserving document structure and formulas with citation markers removed; (ii) full-page screenshots of every figure and table; and (iii) a section-tagged list of hyperlinks categorized as data, code, supplementary material, or other, with surrounding context.

Three-level filtering. We then apply three filtering levels, each targeting a distinct feasibility dimension: task extractability, evaluation automatability, and data completeness.

- **Level 1: Task.** The paper's core contribution must yield an extractable ML task: an algorithmic innovation, an ML formulation of a scientific problem, or a domain adaptation of an established method. We exclude papers in which ML serves only as an auxiliary tool, non-computational studies (wet-lab experiments, pure theory, hardware), and tasks that require physical interaction.
- **Level 2: Evaluation.** The paper must claim state-of-the-art performance on a quality-related metric, rather than on speed, cost, or interpretability. Moreover, this metric must admit a deterministic, fully automated evaluation that does not rely on human judgment, external service dependencies, or components of the algorithm itself.
- **Level 3: Data.** The paper must provide complete data for at least one evaluation instance: an experimental setting with a development set D_{dev}, an evaluation set D_{eval}, and corresponding evaluation metrics. These data must match the version used in the paper and be publicly accessible without application or authentication. We also check whether the evaluation set can be separated into test inputs X_{test} and reference answers Y_{ref}. We further tag each dataset by volume (Tier S $<$ 1 GB, Tier M 1–50 GB, Tier L $>$ 50 GB) and reject papers whose data exceeds 50 GB.

B.2 Dataset Acquisition and Verification

This subsection details the dataset acquisition and verification procedures outlined in §2.3.

Dataset acquisition. We clone the linked code and data repositories and download the datasets by size tier and priority, taking the evaluation instances behind the paper's main results first. Tier S datasets are downloaded in full, while Tier M datasets are downloaded one instance at a time under a cumulative size cap, and we skip the remaining instances once the cap is reached. Tier L papers have already been removed during filtering.

File-level firewall. To keep the information firewall intact, the agent must start exactly where the core algorithm A starts, so it receives the inputs to A but none of A's operations or outputs. We decide which files to keep by one question: *is this file needed to define the task or provide the initial conditions for solving it?* Files that define the task or provide legitimate initial inputs and resources are retained, including raw inputs that precede A, outputs of task-defining data preparation shared across methods, and external resources on which A depends. Files that contain A's own operations or outputs are excluded, including outputs of preprocessing specific to A and its intermediate or final outputs, along with any irrelevant files. We make each decision by reading the paper, the code, and the materialized data together.

Dataset verification and review. The filter judges feasibility from metadata alone, so we now re-run checks on the downloaded files. Two properties matter most. *Decomposability*: whether D_{dev} separates from D_{eval} using only sample-level splits and method-agnostic preparation (no algorithm or evaluation-time operations), and whether X_{test} separates from Y_{ref} while preserving all available features. We rate each split's difficulty and reject infeasible cases. At this stage we only record the required split procedure. The actual partitioning is performed in §2.4. *Instance validity*: whether the retained evaluation instances correspond to a single research objective and include the core experiment. Non-core or analysis-only instances are discarded. The check succeeds as long as at least one instance is complete. A separate read-only review then cross-references the paper, code, and files to re-verify the A-boundary and all recorded descriptions. A fix step then repairs the record and reconciles the directory by removing surplus or leaking files and re-acquiring missing components, so that both the record and the data are ready for task construction. Cases with extensive corrections are confirmed by manual review.

B.3 Task Package Construction

This subsection details the task package construction procedures outlined in §2.4. Table 3 summarizes the package structure.

Data organization. Following the decomposition procedure from §2.3, we route inputs to the agent-visible `problem/data/` and reference answers to the hidden `evaluation/ground_truth/`, with the routing rule determined by the reference-answer

Table 3: **Task package structure produced by NatureGym.** Components under `problem/` are agent-visible; those under `evaluation/` are hidden from the agent.

Visibility	Component	Contents
Agent-visible	`problem/README.md`	Task definition, evaluation metrics, output format, submission specification
	`problem/data_description.md`	Dataset overview, file formats and schemas
	`problem/data/`	Per-instance inputs (*ground truth excluded*)
Hidden	`evaluation/evaluator.py`	Deterministic scoring function with input validation
	`evaluation/ground_truth/`	Per-instance reference answers
Infrastructure	`environment/Dockerfile`	Per-task overlay on the shared base image
	`metadata.json`	Domain, compute requirements, per-instance SOTA scores

type (static label, oracle function, or distributional statistic). Instances whose required evaluation components cannot be sourced from public libraries or reimplemented from author code are excluded. Construction continues as long as at least one instance remains viable.

Task documentation. Each package ships two documents under the information-firewall constraint. `data_description.md` is a technical reference for the files in `problem/data/`, covering dataset overview, formats, and schemas. `README.md` defines the task, evaluation metrics, output format, and submission specification, retaining only the quality metrics the paper uses for ranking and designating one primary metric per instance for aggregate scoring. `metadata.json` records the scientific domain, compute requirements, and per-instance SOTA scores extracted from the paper text, tables, or figures.

Automated evaluator. The evaluator independently scores agent outputs, dispatching on the reference-answer type: it compares against the ground truth for *Label* tasks, runs the scoring function for *Oracle* tasks, and computes distributional statistics for *Distribution* tasks. It validates output format and shape before scoring, and scores multi-instance tasks with failures isolated so that one does not affect the rest. We check the evaluator at build time with logic tests, smoke tests, comparison against author code where available, and verification of evaluator scores against the paper's reported values using the authors' released outputs.

Execution environment. A shared base image pre-installs core scientific and ML libraries. Task-specific dependencies are layered on top via per-task Dockerfiles, with a standalone build reserved for irreconcilable conflicts such as a different CUDA or Python version.

B.4 Package and Environment Review

Unlike the one-shot reviews of the filtering and data-acquisition stages, this review runs a verify–repair loop that iterates until the final artifact is structurally complete, internally consistent, stably scorable by the evaluator, and buildable into a working environment, all while preserving the information firewall. It has three parts.

Build-time self-audit. Before completing the construction, a final step re-reads the paper and the structured record to recheck the task definition, data alignment, metadata tags, SOTA scores, and the firewall. Anything the automated process is uncertain about is flagged for human review before the loop proceeds.

Task-package verification. We run 36 checks across five dimensions: artifact completeness, cross-component consistency, the information firewall, benchmark-design principles, and end-to-end dynamic testing. The dynamic test runs a simple baseline solver that follows the `README` interface end to end over all instances and feeds its outputs to the evaluator, checking that the score structure and values are sensible, and adds a correctness test (ground truth as a perfect prediction

should score near-perfect) and a robustness test (malformed inputs must fail cleanly rather than yield spurious scores). Failed checks are graded by severity and trigger minimal targeted repairs. After each repair we immediately re-run the relevant consistency scans and dynamic tests to confirm that the repair itself introduces no new error. This verify–repair cycle iterates over multiple rounds until the verification passes, and what cannot be reliably auto-repaired is escalated to human review.

Environment verification. We build the Docker image on a physical machine, run library imports and verify that library versions match our presets. When a build fails, we separate root causes from cascading symptoms and classify each root cause by type. Repair follows one core principle: never override a base-image package. Working from least to most disruptive, we (i) switch to a base-compatible version, (ii) add the missing dependency or runtime configuration, (iii) substitute a compatible alternative and rewrite the affected code, or (iv) remove non-essential packages. A task-critical dependency that resists all of the above triggers a standalone Dockerfile that does not inherit the shared base. Throughout, evaluator and solver dependencies are treated as mandatory and domain convenience packages are best-effort. This verify–diagnose–repair cycle repeats until all checks pass.

C Source Corpus Details

This appendix expands the source-corpus selection summarized in §3.1.

We select source journals by three criteria. First, accepted papers must contain concrete algorithmic contributions with numerical SOTA claims, providing a clear competition target for each task. Second, the journal must include papers with available data, so that the underlying datasets are publicly recoverable without per-item manual approval. Third, the journal's topical scope must cover scientific machine learning, the domain where automated-agent capability is least studied. Accordingly, we select ten Nature-family journals: *Nature Machine Intelligence*, *Nature Communications*, *Nature Methods*, *Nature Materials*, *Nature Biomedical Engineering*, *Nature Energy*, *Nature Biotechnology*, *Nature Computational Science*, *Nature Genetics*, and *Nature Neuroscience*. The publication window is 2022–2025, chosen to balance corpus size against software-stack currency and data-contamination risk. The final 90-task set draws from six of these journals. The other four retain no tasks after filtering, data verification, task construction, and calibration.

Table 4: **NatureGym pipeline funnel,** grouped into five phases aligned with the pipeline of §2. Counts marked with "∼" are approximate, and only the final corpus size is exact.

Stage	Step	Papers retained
Collection	Initial crawl from 10 Nature-family journals	∼5,500
Filtering	Article-type filter (exclude non-research)	∼2,500
Filtering	Three-level filtering (§2.2)	∼200
Acquisition	Dataset acquisition and verification (§2.3)	∼180
Construction	Task construction (§2.4)	∼160
Calibration	Evaluation-time quality calibration (§3.2)	**90**

D Benchmark Quality Calibration Details

This appendix expands the benchmark quality calibration summarized in §3.2.

First-round diagnosis categories. The exposed defects fall into six categories: (1) ground-truth leakage, where the test input carries an unintended channel that allows the agent to recover the answer; (2) distorted task definitions, where the target degenerates into a deterministic function of input features and can be exactly solved rather than learned; (3) metrics that fail to distinguish shortcuts from genuine solutions; (4) evaluator or anchor inconsistencies (e.g., an evaluator metric that disagrees with the task description or metadata); (5) pipeline or environment errors; and (6) missing data resources. Locally verifiable defects receive minimal targeted repairs. Mitigable risks are recorded and backstopped by the information firewall, the web-search-disabled container, and the validity judge. Tasks with broken definitions, unverifiable metrics, or irreparable leakage

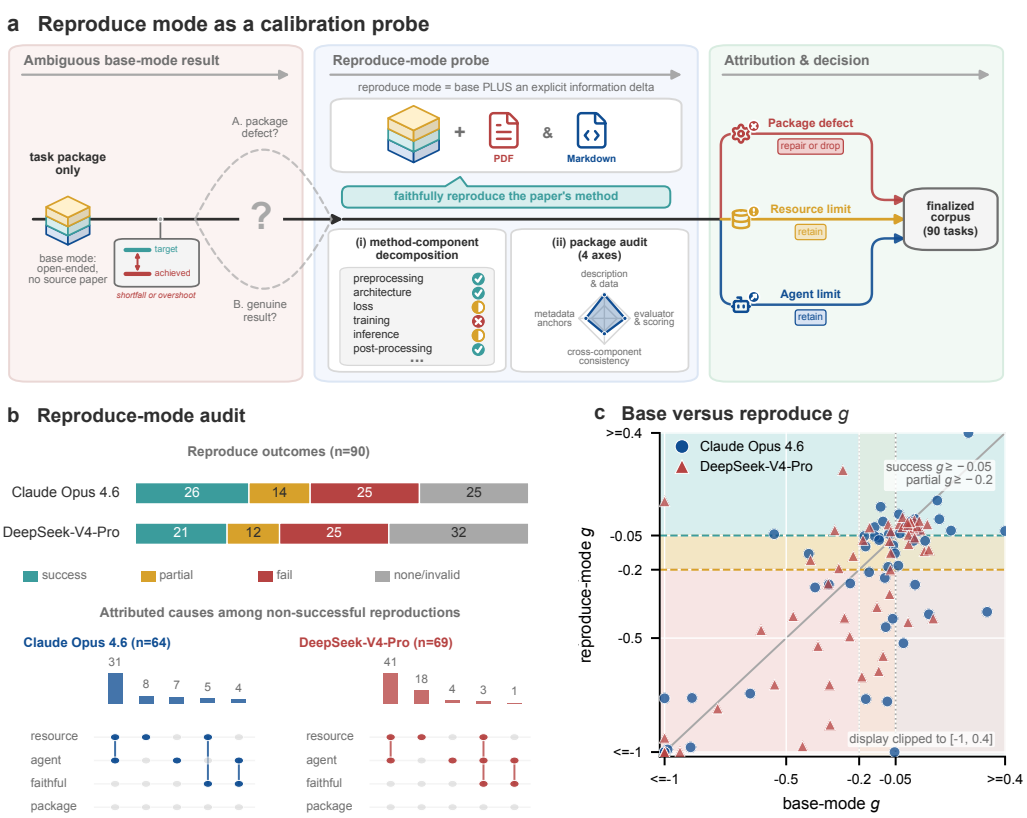

Figure 6: **Calibrating the NatureBench corpus.** **a**, The reproduce-mode calibration probe, from an ambiguous result to a clear attribution. **b**, Reproduce-mode outcomes and attributed causes for Claude Opus 4.6 and DeepSeek-V4-Pro, showing five highest-frequency cause combinations. **c**, Per-task g in base versus reproduce mode.

are dropped. Runs that are legitimate but low-scoring, timed-out, or judged invalid are retained as normal agent failures.

Reproduction-mode audit procedure. For each case in reproduce mode, we decompose the paper's method into components (e.g., preprocessing, architecture, loss, training, inference, post-processing), rate each as full, partial, or missing, classify the score outcome, and attribute any anomaly to the agent, the runtime resources, or the package. Regardless of score, we audit package quality along four axes: (1) task description and data, (2) evaluator and scoring, (3) metadata anchors, and (4) cross-component consistency. For example, we check whether the SOTA anchor is drawn from the same dataset and granularity as the evaluator computes, whether metadata and evaluator scores share the same scale and units, whether required training data and external resources are present, whether the evaluator returns a reasonable score on ground truth, and whether the task description is consistent with the paper's method. After human review, 45 tasks are dropped for defects that would systematically contaminate the main evaluation (missing data, evaluator deviations, absent required information, leakage, or distorted scoring), and 17 tasks receive minor repairs (e.g., anchor-value alignment, scale reconciliation, evaluator-logic corrections, incomplete-instance removal, environment and serialization fixes).

Reproducibility analysis. On the finalized 90 tasks, we quantify SOTA-anchor attainability (success: $g \geq -0.05$; partial: $-0.2 \leq g < -0.05$). Claude Opus 4.6 reproduces 26/90 tasks successfully and 14/90 partially; DeepSeek-V4-Pro reproduces 21/90 and 12/90. At least one model succeeds on 33/90. Both succeed on 14/90, where g clusters tightly around zero (median -0.0026, 90% of absolute deviations ≤ 0.034). Reproduce-mode success is lower than base mode: Opus drops from $32/90$ to $26/90$ and DeepSeek from $25/90$ to $21/90$, primarily because faithful re-

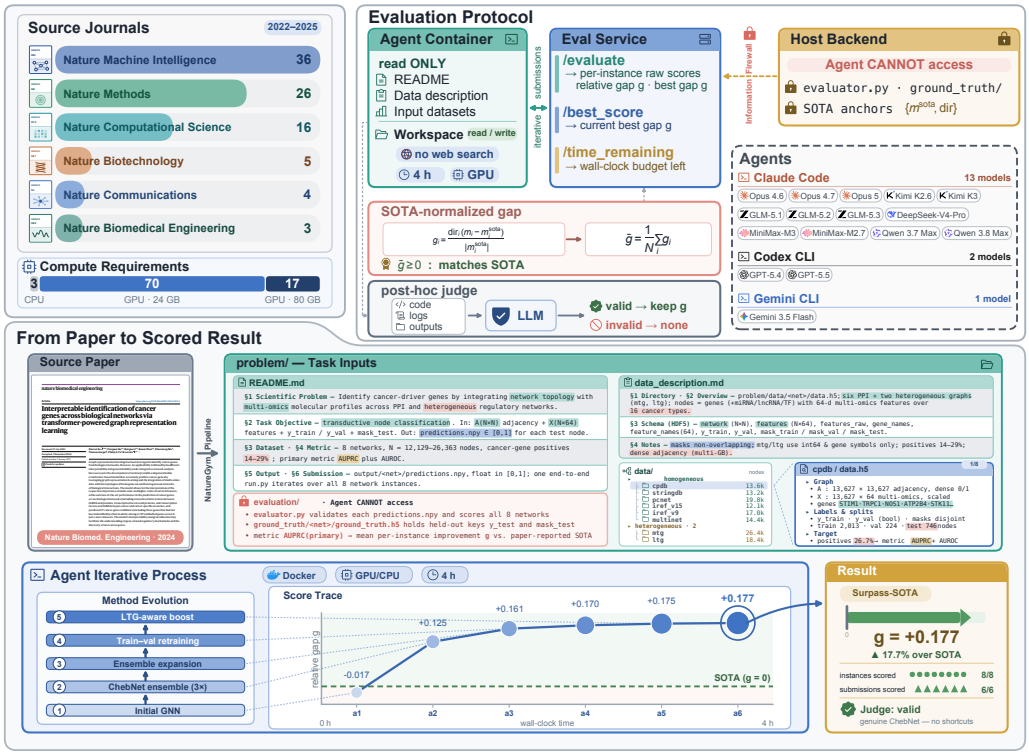

Figure 7: **NatureBench task construction and evaluation pipeline.** Tasks are constructed from Nature-family papers using NatureGym. Agents solve them in isolated containers, while a host-side service scores submissions with hidden evaluators, and a post-hoc judge screens run validity. The example shows iterative submissions reaching a valid SOTA-surpassing result.

production triggers heavier training and more complex dependencies. DeepSeek's no-result count rises from 1 to 29, accounting for most of its gap. Root-cause attribution of non-success cases is dominated by insufficient compute or time and method simplification rather than package defects.

E Benchmark Statistics Details

This appendix expands the scientific coverage and evaluation design summarized in §3.3.

Breadth and representativeness. Figure 3 summarizes NatureBench coverage along three single-label axes (source journal, scientific domain, and ML task type) together with a multi-label view of the source papers' contribution nature. By provenance, the final 90 tasks concentrate in six journals, led by *Nature Machine Intelligence* (36), *Nature Methods* (26), and *Nature Computational Science* (16). The corpus skews recent, with 11, 17, 28, and 34 tasks for 2022 through 2025. Across scientific domains, the tasks span six areas (cellular omics, protein biology, biomedical modeling, physical modeling, molecular design, and relational reasoning) and eight ML task types, where prediction/regression and classification dominate, followed by clustering/integration and a long tail of generation, segmentation, simulation, structure-modeling, and other specialized tasks.

Heterogeneous evaluation design. Figure 3 reports the design summary. At the *Scope* layer, tasks are evaluated over multiple instances (mean 3.7, median 3, up to 19), organized under varied data-partition topologies: most use multiple independent test sets, but many use a shared training set with multiple test sets or leave-one-out cross-dataset splits, so evaluation extends beyond a single dataset to generalization conditions. Agent-visible data ranges from under 1 GB (about half the tasks) to over 10 GB (about a fifth). By primary input modality, the tasks span biological sequences, molecular and materials structures, single-cell and spatial omics, imaging and volumetric

data, temporal signals and spectra, graphs and networks, and feature tables. At the *Paradigm* layer, most tasks use a static label scored against hidden ground truth. The remaining tasks are either *distribution* tasks, where the agent generates samples scored by set-level or distributional metrics, or *oracle* tasks, where the agent optimizes against a provided scorer with no fixed correct answer. At the *Metric* layer, the tasks use 80 distinct primary metrics (AUROC, RMSE, Spearman ρ, ARI, F1, MAE, among others), with each task typically scored by several (mean 3.7 primary, 5.1 auxiliary), most of which are higher-is-better.

F Evaluation Protocol Details

This appendix details the execution environment, service interfaces, and adjudication summarized in §3.4.

The agent operates inside an isolated, task-specific Docker container with read access to `problem/` (task description and data) and read/write access to `workspace/`, a 4-hour wall-clock budget, and one GPU when the task requires it. The evaluator, ground truth, and SOTA target reside in a host-side evaluation service that the agent cannot access directly. *During* the run, the agent iteratively queries this service through three endpoints. `/evaluate` scores a submission on every instance across all reported metrics and returns raw scores, relative gaps, and the running best. `/best_score` returns the current best without submitting. `/time_remaining` reports the remaining budget. The wall clock pauses during scoring so that evaluation overhead does not consume the agent's budget. *After* the run, a post-hoc GPT-5.5 judge checks for shortcut behavior (output fabrication, rule substitution for learning, answer recovery, feedback gaming, or training bypass) and assigns flagged runs a score of `none` (Figure 7).

G Supplementary Performance Statistics

Table 5 supplements the main results in §4.2 with gap statistics and submission rates for all 16 agents.

Table 5: **Gap summary and submission rates of agents on NatureBench.** The SOTA-normalized gap g (§3.4) is summarized by its median $\tilde{g}$ and mean $\bar{g}$; the $\cdot_{\text{all}}$ columns set $g = -1.0$ for tasks with no valid score, while the $\cdot_{\text{valid}}$ columns cover only judge-accepted tasks. **CR** (Completion Rate) and **SR** (Score Rate) are the fractions of tasks yielding a valid score and any score. Best/second per column in **bold**/underlined.

		Gap Summary				Submission Rates (%)	
Model	Harness	$\tilde{g}_{\text{all}}$	$\bar{g}_{\text{all}}$	$\tilde{g}_{\text{valid}}$	$\bar{g}_{\text{valid}}$	CR	SR
Claude Opus 5	Claude Code	**0.014**	**−0.19**	**0.027**	**−0.08**	87.8	98.9
Claude Opus 4.7	Claude Code	−0.030	−4.58	−0.007	−5.13	86.7	**100.0**
Kimi K3	Claude Code	−0.127	−4.12	−0.036	−4.96	78.9	97.8
GLM-5.2	Claude Code	−0.081	−5.45	−0.058	−6.00	**88.9**	98.9
Qwen 3.8 Max	Claude Code	−0.078	−1.99	−0.023	−2.20	82.2	**100.0**
Claude Opus 4.6	Claude Code	−0.118	−2.16	−0.083	−2.37	84.4	**100.0**
GLM-5.3	Claude Code	−0.300	−0.88	−0.055	−0.82	66.7	95.6
Gemini 3.5 Flash	Gemini CLI	−0.200	−5.54	−0.084	−7.01	75.6	98.9
Kimi K2.6	Claude Code	−0.169	−10.18	−0.085	−12.16	82.2	94.4
Qwen 3.7 Max	Claude Code	−0.205	−2.74	−0.111	−3.18	80.0	98.9
MiniMax-M3	Claude Code	−0.160	−6.03	−0.114	−6.80	86.7	98.9
GLM-5.1	Claude Code	−0.197	−8.56	−0.116	−10.58	78.9	93.3
GPT-5.4	Codex CLI	−0.172	−3.59	−0.106	−4.28	78.9	**100.0**
GPT-5.5	Codex CLI	−0.462	−0.52	−0.053	−0.12	54.4	98.9
DeepSeek-V4-Pro	Claude Code	−0.292	−8.63	−0.225	−9.70	87.8	98.9
MiniMax-M2.7	Claude Code	−0.501	−8.82	−0.301	−10.26	84.4	98.9

Figure 8 shows the corresponding gap distributions and mean and median gaps.

H Validity and Scope of Performance Estimates

This appendix expands the validity and scope discussion in §4.2.

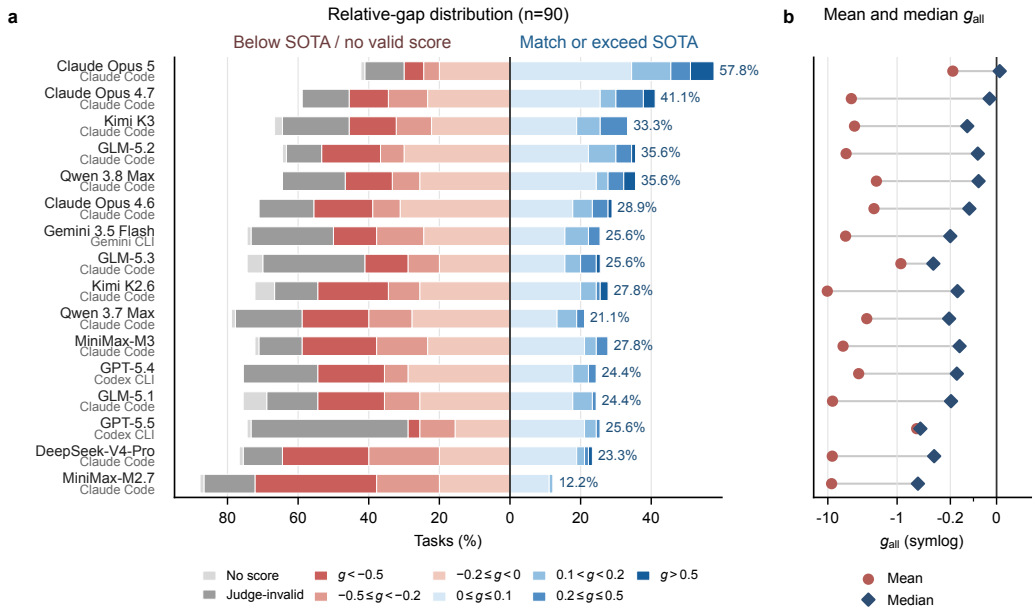

Figure 8: **Gap distribution and summary of agents on NatureBench. a**, Percentage of tasks in each g interval for each agent, arranged around the SOTA target ($g = 0$). **b**, Mean and median relative gap over all tasks (g_{all}), assigning $g = -1.0$ when no valid score is available.

NatureBench converts public papers into automatically scored tasks and normalizes their heterogeneous metrics onto a common SOTA-relative scale. To verify that this design does not distort the results, we audit the tasks with extreme scores and those most exposed to leakage or gaming. We examined each concern and found it either working as designed or bounded to acceptable levels by the protocol.

Metric normalization. Extreme scores are a property of the SOTA-relative metric rather than a sign of a faulty task, surfacing as the heavy negative tail in Figure 8 and the gap between $\bar{g}$ and $\tilde{g}$ in Table 5. The gap $g = (\text{score} - \text{SOTA})/|\text{SOTA}|$ scores each result as a fraction of the reported SOTA, so its magnitude depends on that SOTA as much as on the agent. A SOTA score close to zero leaves a tiny denominator, so a merely moderate agent maps to a large negative g on a genuinely hard task. A large positive g may arise where the single primary metric used for scoring captures only one facet of a multi-objective method that its source paper evaluates with several metrics across different aspects: an agent optimizing for it directly can exceed the reported value without pursuing the method's other objectives. Auditing every extreme-gap task, we find no task error. We therefore use Surpass- and Match-SOTA as the primary metrics and the median $\tilde{g}$ as a tail-robust summary, with the mean only for completeness.

Task coverage. Some tasks evaluate only a bounded slice of their source paper, an unavoidable and reasonable narrowing. Each such task retains the paper's core quantitative problem and scores a subset of instances and metrics. When the omitted instances or metrics cover other directions of the contribution, the paper is captured only in part. A direction is usually excluded because it cannot be captured as structured data or scored automatically and deterministically. Separately, obtainable instances past a task's data-volume budget are also not collected. The retained slice is still the paper's core quantitative task and is scored correctly, so Surpass- and Match-SOTA measure performance on that slice, not on the whole paper.

Leakage and feedback. The residual leakage and feedback risks are unavoidable but constrained by the protocol and confirmed bounded by review. Because tasks are built from public data, some information is in principle accessible: source datasets come from public repositories and benchmarks, and on a few tasks the agent-visible inputs are inherently coupled to their targets, so an agent might read off part of the answer rather than compute it. A secondary risk is that exact-score feedback over

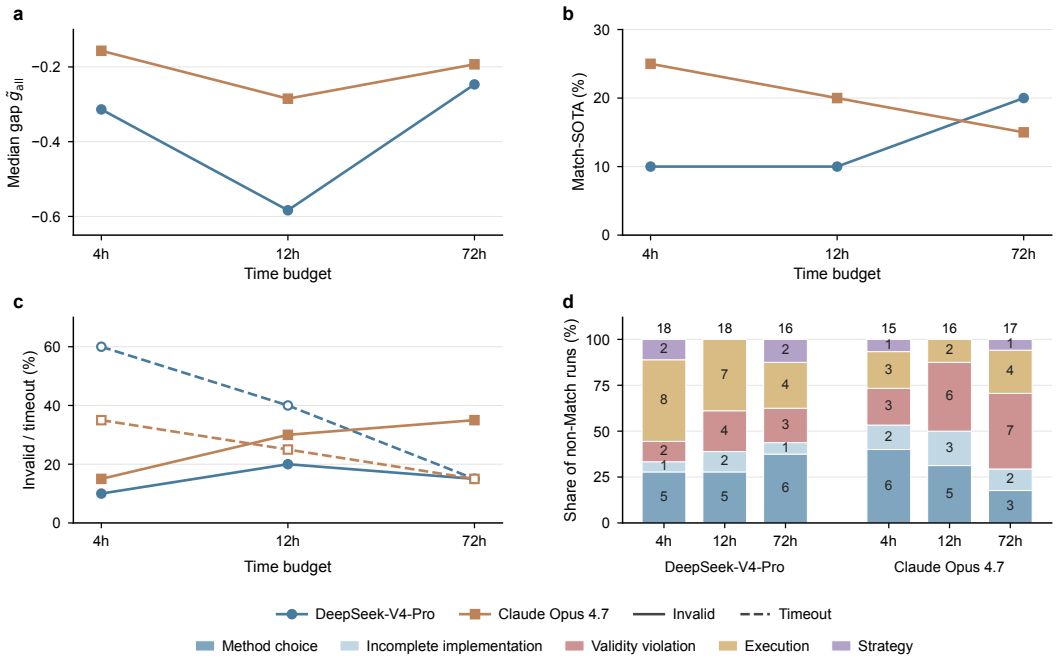

Figure 9: **Time scaling on 20 NatureBench tasks.** **a**, Median gap $\tilde{g}_{\text{all}}$. **b**, Match-SOTA rate. **c**, Invalid and timeout rates. **d**, Failure categories among non-Match runs. Method-layer failures are split into method choice, incomplete implementation, and validity violations; execution- and strategy-layer failures remain aggregated.

repeated submissions lets an agent game the scorer rather than solve the task. The protocol bounds both risks by disabling web search and using a post-hoc validity judge to safeguard the validity of the reported results (the SR–CR gap in Table 5). Reviewing the most at-risk tasks, we find high-frequency submission is overwhelmingly legitimate iteration, and the rare genuine exploit is caught by the judge.

I TIME SCALING

The failure analysis in §5.1 attributes some failures to insufficient time. We therefore test whether more time for discovery and iteration improves Match-SOTA rates. We select 20 tasks across six domains: 16 with budget-related signals in the earlier multi-model analysis and four near-SOTA controls. We evaluate DeepSeek-V4-Pro and Claude Opus 4.7, representing lower and higher performance in the main evaluation, both through Claude Code. We compare separate runs with 4h, 12h, and 72h budgets, retaining the main evaluation's scoring and validity criteria. All triplets below follow this budget order.

Across both models, timeouts decrease from 19 to 13 to 6 out of 40 runs per budget, while matches are 7, 6, and 7 (Figure 9). For DeepSeek-V4-Pro, matches are 2, 2, and 4 out of 20; invalid runs are 2, 4, and 3; and median gaps $\tilde{g}_{\text{all}}$ are -0.314, -0.584, and -0.247. For Claude Opus 4.7, the corresponding counts are 5, 4, and 3 matches and 3, 6, and 7 invalid runs, with median gaps of -0.157, -0.285, and -0.193.

The decline in timeouts suggests that larger budgets relieve some time pressure on discovery and optimization. Using the failure taxonomy in §5.1, we classify 22 of the 33 non-Match runs at 72h as method-layer failures, including nine method-choice failures and ten validity violations. Execution-layer failures persist in eight runs, including four budget-related failures. Method suitability and validity remain major obstacles under the larger budget.

DeepSeek-V4-Pro shows stronger signs of budget constraints than Claude Opus 4.7, with more timeouts and budget-related failures at 4h. At 72h, its budget-related failures decrease from seven to

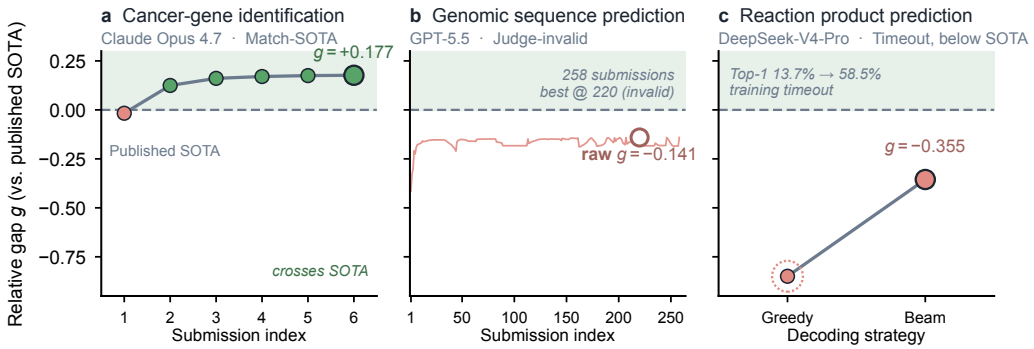

Figure 10: **Representative agent trajectories in NatureBench.** **(a)** Cancer-gene identification (Claude Opus 4.7), six submissions. **(b)** Genomic sequence prediction (GPT-5.5), best at attempt 220; the run is judged invalid. **(c)** Reaction product prediction (DeepSeek-V4-Pro).

three and its Match-SOTA rate rises from 10% to 20%. Nevertheless, overall performance remains limited: 16 of 20 runs at 72h fail to match SOTA, including six method-choice failures and three validity violations. By contrast, Claude Opus 4.7 shows more invalid solutions and fewer matches under larger budgets. Invalid runs increase from three to six to seven, while matches decrease from five to four to three. At 72h, validity violations account for seven of 17 failures, the largest individual subtype.

These results indicate that time limits contribute to some failures but do not fully explain the performance gap. Although budget-related failures decrease with larger budgets, most runs still fail to match SOTA, with persistent method-choice and validity problems. On these selected tasks, extending the budget alone is insufficient to deliver consistent gains in valid performance.

J Case Studies

This appendix complements the aggregate analysis in §5.1 with three representative agent trajectories. The cases cover three recurring outcomes in NatureBench: a method-aligned solution that matches SOTA, an invalid solution with hand-written rules, and a plausible long-horizon solution limited by execution depth. All cases are drawn from the final 90-task, 16-agent analysis used in §5. Table 6 lists the selected cases and Table 7 summarizes their trajectory-level mechanisms, while two figures show how each case plays out. Figure 10 traces each agent's score across its submission sequence, and Figure 11 decomposes the best submission of the two multi-instance cases into per-instance gaps, showing where the aggregate score comes from.

Table 6: Representative trajectory cases analyzed in Appendix J.

Case	Agent	g	Status
Cancer gene identification on biological networks	Claude Opus 4.7	0.17666	Match-SOTA
Genomic sequence prediction	GPT-5.5	raw −0.14087	Judge-invalid
Organic reaction product prediction	DeepSeek-V4-Pro	−0.35540	Timeout, below SOTA

Case 1: method-aligned graph modeling can produce a valid success. The first task is derived from TREE, a transformer-powered graph representation learning study for identification of cancer-genes (Su et al., 2025). It asks the agent to identify cancer-associated genes on eight biological networks. Each instance provides a network adjacency matrix, 64-dimensional multi-omics node features, training and validation labels, and a test-node mask. The source problem is naturally a graph-based binary node-classification problem: its core scientific objective is to combine biological network structure with multi-omics node attributes to prioritize cancer genes. The primary metric is AUPRC on each network, aggregated as improvement relative to the paper-side SOTA.

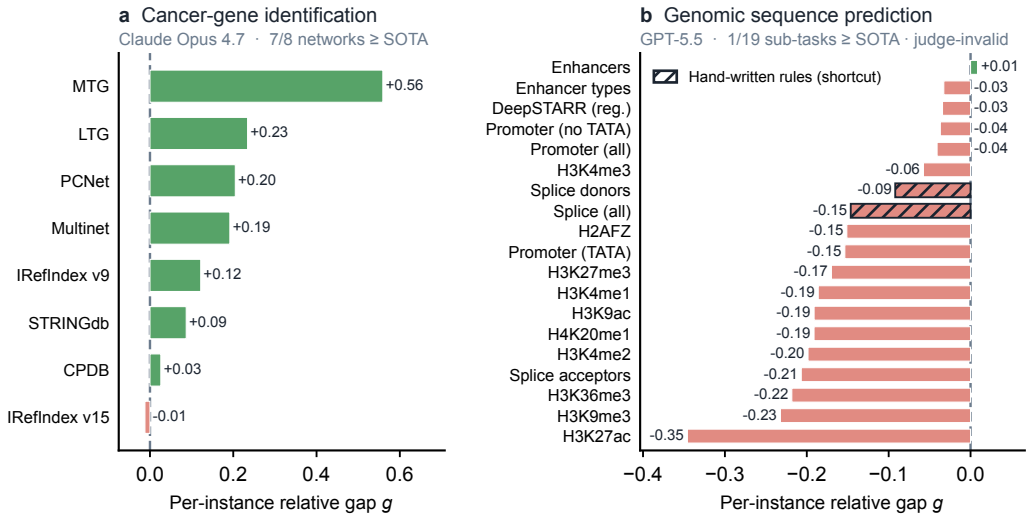

Figure 11: **Per-instance relative gap g at each agent's best attempt. (a)** Cancer-gene identification (Claude Opus 4.7), eight biological networks. **(b)** Genomic sequence prediction (GPT-5.5), 19 genomic sub-tasks; hatching marks the two hand-written-rule outputs that triggered invalidation. The single-instance reaction task is omitted.

Claude Opus 4.7 selected a route that matched this task structure. The final solution implements a Chebyshev polynomial graph convolutional network (ChebNet) ensemble: it loads the HDF5 network data and node features, computes normalized graph Laplacians, trains with validation AUPRC early stopping, then retrains on the combined train and validation labels before averaging models across Chebyshev orders, depths, and random seeds. The judge marked the submission valid because the predictions were generated by trained graph models, and the raw logs show progressive AUPRC improvements across submissions. Table 8 summarizes the score progression for this trajectory.

This is a genuine agent success. It correctly treated the task as graph-based node classification and used an appropriate GNN, class-imbalance handling, validation-based early stopping, and ensembling to push the score above SOTA. The per-instance results were also uneven: MTG, LTG, PCNet, and Multinet improved substantially, whereas IRef v15 remained slightly below the paper-side SOTA. However, from another aspect, the agent did not propose a new method of cancer-gene identification.

Case 2: hand-written splice-site rules render the run invalid. The second task is derived from the Nucleotide Transformer benchmark for human genomics (Dalla-Torre et al., 2025). It contains 19 genomic sequence prediction instances, spanning histone marks, enhancers, promoters, splice sites, and enhancer-activity regression. The source paper's core idea is to learn broad DNA sequence representations from large-scale pretraining and transfer them to diverse downstream sequence-function tasks. The agent must submit predictions for 18 classification tasks and one regression task.

Table 7: High-level mechanisms observed in the representative trajectories.

Case	Agent route	Outcome driver
Cancer gene identification on biological networks	ChebNet/GNN ensemble	Method alignment and training optimization
Genomic sequence prediction	Fitted sequence models plus two hand-written splice-rule outputs	Hand-written prediction rules
Organic reaction product prediction	Seq2seq reaction modeling	Insufficient execution depth

Table 8: Score progression and diagnosis for the cancer-gene identification case.

Stage	Evidence	Diagnosis
Initial graph model	$g = -0.01715$	The first submission was runnable, but still slightly below SOTA.
First crossing	$g = 0.12457$	Once the graph-modeling route matured, most networks improved substantially.
Ensembling and training optimization	$g = 0.161$ to 0.175	Chebyshev order, depth, random seeds, and train-plus-validation retraining continued to add gains.
Final strengthening	$g = 0.17666$	The last round mainly improved the LTG network and produced the best aggregate score.

This trajectory is long and technically substantial. The agent produced 258 submissions, with the best raw score at attempt 220. It began with compact k-mer count models and fast linear classifiers, then added splice-site motif rules, GPU CNNs, enhancer-activity CNN ensembles, a two-stage enhancer-type classifier, and many threshold sweeps. However, the final solution generated `splice_sites_all` and `splice_sites_donors` using hand-written score functions and fixed cutoffs rather than learned or fitted behavior. The judge identified this shortcut and marked the run invalid. Table 9 summarizes the main trajectory stages.

Table 9: Score progression and diagnosis for the genomic sequence prediction case.

Stage	Evidence	Diagnosis
Fast baseline	raw $g = -0.41445$	The agent first solved the submission-completeness problem.
Task specialization	raw $g = -0.20882$	Trained models improved performance on several sub-tasks.
Final raw solution	raw $g = -0.14087$	Valid fitted-model tuning coexisted with hand-written rules for two splice-site outputs.
Validity audit	Judge-invalid	The identified shortcut invalidated the score.

The failure is therefore methodological validity. The agent built and iterated fitted models for most sub-tasks, but two outputs relied on hand-written predictive rules. This case shows that a localized shortcut can invalidate a run even when most outputs come from fitted models.

Case 3: a plausible route can be limited by execution depth. The third task is derived from LocalTransform, a generalized-template-based graph neural network for organic reactivity prediction (Chen & Jung, 2022). It asks the agent to predict major organic reaction products for USPTO-480k atom-mapped reactants. The source paper's core idea is to model local reaction centers and bond changes with reaction templates, molecular graph representations, and chemistry tooling. This route reaches a Top-1 exact-match accuracy of 0.908, while a strong sequence-to-sequence baseline reaches 0.887. The task requires both learning reaction transformations from hundreds of thousands of examples and generating ranked product SMILES efficiently.

The agent selected a plausible but expensive route: it implemented a complete sequence-to-sequence reaction model with a SMILES tokenizer, dataset loader, Transformer model, training loop, checkpointing, and prediction pipeline. The judge marked the submission valid because the final predictions were generated by the trained model with checkpointing and beam-search inference. Table 10 summarizes the trajectory stages and score progression.

The key limitation was execution depth rather than invalidity. The agent found a reasonable scientific-computational route, but the task required deeper training, more efficient generation, and more specialized chemical modeling than the fixed budget allowed. This case illustrates the execution-layer failures discussed in §5.1: some agents identify a plausible direction, yet fail because the required training and inference loop is too long.

Table 10: Score progression and diagnosis for the reaction product prediction case.

Stage	Evidence	Diagnosis
Route selection	24.3M parameters	The route was plausible, but computationally heavy.
Long training	Loss $2.0312 \rightarrow 1.3149$	The model learned, but training consumed much of the budget.
Greedy decoding	Top-1 13.68%	The first valid submission used a weak decoding strategy.
Beam search	Top-1 58.53% $g = -0.35540$	Inference engineering helped sharply, but the final score remained below SOTA.

K Resource Usage Details

Following common practice in agent benchmarks, we report resource usage at the trajectory level rather than only at the initial prompt level. For each evaluated agent, we aggregate token-usage information from usable execution logs and summarize per-case input tokens, output tokens, and estimated cost. Input tokens use exact harness- or provider-reported usage fields. When the log records cache accounting, we retain the distinction among non-cached input, cache read or hit tokens, and cache creation or write tokens; the mean input-token column reports their sum so that it reflects the full amount of context processed during the run.

Output-token accounting follows the most reliable source available for each model. For Claude Opus, GPT, and Gemini runs, we use exact provider- or harness-reported output-token fields. For third-party models executed through Claude Code, the logged output-token fields are incomplete, so we estimate output tokens from agent-authored trajectory text using the standard rule of thumb that one token corresponds to roughly four English characters (OpenAI, 2026c; Anthropic, 2026d). Rows that use this output-token estimate, and the costs derived from it, are marked with an asterisk. Cost is computed with official standard list prices and provider-specific cache rates. We exclude limited-time promotions, batch/flex/priority modes, regional or data-residency uplifts, cache-storage charges that cannot be recovered from the logs, and OpenAI long-context multipliers whose per-request triggers cannot be recovered from aggregate Codex logs. Table 11 reports the resulting per-agent means over runs with valid usage records.

Table 11: Per-agent mean token usage and estimated cost over runs with valid usage records. *Tokens estimated from agent-authored trajectory text and *costs derived from these estimates.

Agent	**Mean input tokens**	**Mean output tokens**	**Mean cost (USD)**
Claude Opus 5	17.04M	126.5K	$19.52
Claude Opus 4.7	24.35M	143.3K	$20.93
Claude Opus 4.6	9.36M	85.7K	$14.40
GPT-5.5	8.25M	31.9K	$6.01
GPT-5.4	10.79M	43.6K	$4.14
Gemini 3.5 Flash	8.72M	36.3K	$3.82
Qwen 3.8 Max	21.44M	166.3K*	$7.22*
Qwen 3.7 Max	13.40M	91.3K*	$10.01*
Kimi K3	10.92M	36.4K*	$4.93*
Kimi K2.6	14.61M	95.1K*	$3.22*
MiniMax-M3	25.83M	100.3K*	$1.82*
MiniMax-M2.7	7.54M	52.1K*	$0.95*
DeepSeek-V4-Pro	12.73M	82.2K*	$0.17*
GLM-5.3	16.83M	101.2K*	$5.20*
GLM-5.2	20.83M	135.9K*	$6.70*
GLM-5.1	11.74M	75.2K*	$8.94*