AUTOSCIENTISTS: Self-Organizing Agent Teams for Long-Running Scientific Experimentation

Ada Fang*
Harvard University
ada_fang@g.harvard.edu

Shanghua Gao*
Harvard University
shanghua_gao@hms.harvard.edu

Marinka Zitnik
Harvard University
marinka@hms.harvard.edu

AUTOSCIENTISTS website: https://autoscientists.openscientist.ai
AUTOSCIENTISTS code: https://github.com/mims-harvard/AutoScientists

Abstract

Scientific research proceeds through iterative cycles of hypothesis generation, experiment design, execution, and revision. AI agents can automate parts of this process, but existing approaches typically follow a single research trajectory or coordinate through a central planner with fixed objectives. As a result, they struggle to sustain parallel exploration, adapt as experimental evidence changes, or preserve knowledge of failed directions over long-running experiments. We introduce AUTOSCIENTISTS, a decentralized team of AI agents for long-running computational scientific experimentation. Agents interpret a shared experimental state, self-organize into teams around promising hypotheses, critique proposals before using experimental compute, and share successes and failures to reduce redundant exploration. Under matched experimental budgets, AUTOSCIENTISTS improves over prior AI agents across biomedical machine learning, language model training optimization, and protein fitness prediction. On BioML-Bench, spanning biomedical imaging, protein engineering, single-cell omics, and drug discovery, AUTOSCIENTISTS achieves a mean leaderboard percentile of 74.4% across 24 tasks, improving over the strongest AI agent by +8.33% in the absolute leaderboard percentile points. On GPT training optimization, AUTOSCIENTISTS reaches a target validation bits-per-byte 1.9× faster than Autoresearch and continues discovering improvements from a starting champion where the single-agent approach finds none (7 vs. 0 accepted improvements). On ProteinGym fitness prediction, AUTOSCIENTISTS discovers a method for ACE2-Spike binding that improves over the current state-of-the-art model by +12.5% in Spearman correlation. Applied without modification across all 217 ProteinGym assays, the same method improves over the prior state of the art by +6.5% (Spearman correlation).

1 Introduction

AI agents for science are beginning to move beyond answering questions and running predefined workflows toward proposing and executing research steps [1], from protein engineering in biology to language model optimization in machine learning [2, 3]. Agents can generate hypotheses, synthesize literature, design computational experiments, write and execute code, and refine models from experimental feedback [4–10]. However, most current approaches remain limited to short horizon

*Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

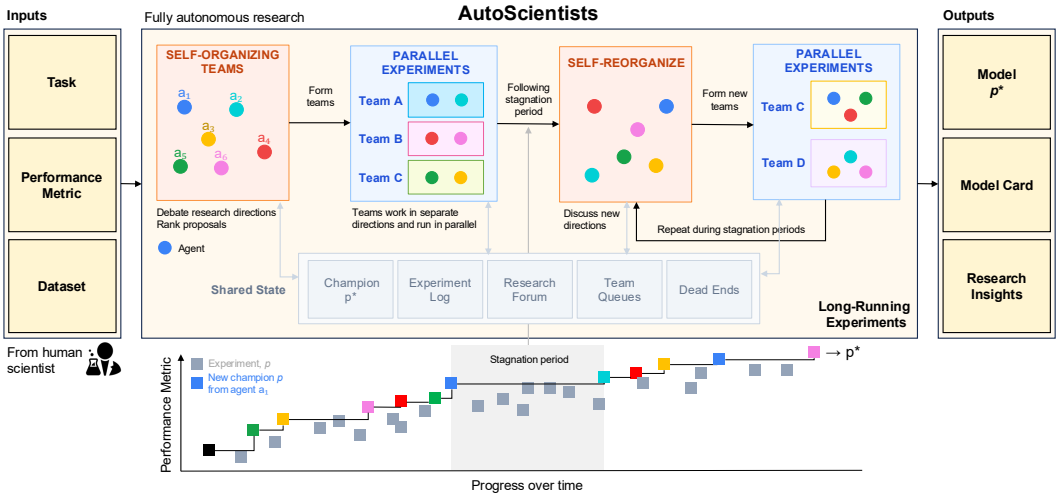


Figure 1: **Self-organizing agent teams for long-running experimentation.** Overview of AUTOSCIENTISTS. Agents identify promising research directions, organize into teams, and execute experiments in parallel.

optimization or fixed pipelines. They typically follow a single reasoning thread or use a search space decomposition set at the start of the run. These designs break down in long-running scientific experimentation, where research directions are not known in advance and change over time.

Existing AI agents can run experiments, but long-running science requires maintaining competing hypotheses, updating them as evidence changes, and using failures to redirect the search. Single-agent systems such as AIDE [11] and Autoresearch [3] iteratively refine proposals but follow a single search trajectory, limiting their ability to explore competing hypotheses in parallel. Multi-agent systems [12–14] distribute work across agents, but still coordinate through a central structure, in which a planner decomposes the problem, a search algorithm ranks proposals, or agents converge through discussion or voting [15, 16]. These approaches assume that the search space can be partitioned into stable directions at the start of the run. *In long-running experimentation, however, productive directions shift as evidence accumulates. Some hypotheses stop yielding improvements, failed directions must be tracked to avoid repeated exploration, and new hypotheses often emerge only after earlier experiments are analyzed.*

Present Work. We introduce AUTOSCIENTISTS, a self-organizing agent team for long-running scientific experimentation that coordinates without a central orchestrator agent (Figure 1). Rather than receiving assignments from a planner, agents act on a shared state that records proposals, experiments, results, failures, and the current champion. Teams form dynamically through agent interaction rather than a user specified decomposition. Agents post experiment proposals to a shared forum, where peers critique them before execution, filtering weak ideas before compute is committed. As results accumulate, agents reorganize around productive directions, retire exhausted directions, and share successes and failures across teams to reduce redundant exploration. We apply AUTOSCIENTISTS to research tasks spanning imaging, drug discovery, single-cell omics, protein engineering, protein fitness prediction, and language model training optimization.

Across benchmarks, AUTOSCIENTISTS improves over existing AI agents. On BioML-Bench [2], AUTOSCIENTISTS achieves the highest average leaderboard percentile among the evaluated agents, reaching 74.40% across 24 biomedical ML tasks compared with 66.07% for Autoresearch under the same task interface, model backend, and hardware budget. The performance improvements are largest in drug discovery, where AUTOSCIENTISTS improves from 46.16% to 64.52%. On GPT nanochat training optimization [3], AUTOSCIENTISTS reaches the same intermediate validation loss in 34 experiments that Autoresearch reaches in 65 experiments, and when continuing from an AUTOSCIENTISTS champion reaches a validation bits-per-byte (bpb) of 0.9730 while Autoresearch finds no accepted improvements over 100 experiments. On ProteinGym supervised substitution fitness prediction [17], AUTOSCIENTISTS starts from Kermut and discovers a Kermut extension that improves ACE2–Spike binding Spearman's ρ from 0.747 to 0.840. Furthermore, the frozen recipe transfers across the full 217-assay ProteinGym supervised substitution benchmark, improving the official average Spearman's ρ from 0.657 to 0.700. Below we summarize our contributions.

- **A self-organizing agent team for long-horizon scientific experimentation**. Unlike prior systems that rely on central coordinators, consensus-based discussion, or fixed decompositions of the search space, AUTOSCIENTISTS allows agents to independently interpret a shared experimental state and decide which hypotheses to pursue. Agents post proposals to a shared forum where peers critique and filter them before experiments run, allowing teams and experimental directions to emerge through interaction rather than external assignment.
- **State-of-the-art performance across scientific domains, with sustained improvement during long-running experimental search.** AUTOSCIENTISTS improves over prior agents on biomedical ML, protein fitness prediction, and language model training optimization, and continues identifying productive modifications after single-agent baselines stop improving.

2 Related Work

AI Agents for Scientific Research. AI agents are being developed to automate scientific workflows, including literature review, hypothesis generation, tool use, code execution, experimental design, benchmarking, and manuscript drafting [18–22]. Biomedical agents combine multi-step reasoning with biomedical tools, literature grounding, omics analysis, code execution, and evidence reconciliation [10, 6, 9, 23–25]. Other systems push toward longer horizon discovery through repeated cycles of literature search, hypothesis generation, debate, refinement, tool integration, optimization, equation discovery, self-directed exploration, and skill accumulation [4, 7, 26–31, 12, 32]. Several scientific agent systems rely on role specialized architectures, such as PI–scientist–critic organizations or Manager–Developer–Critic–Tool Creation pipelines [14, 5]. In AI research, related systems have also generated research papers or evolved algorithms through iterative code modification and experimentation [33, 3, 34]. Our system differs in that agents collectively determine research directions through discussion and coordinate through shared forums rather than fixed pipelines or a central orchestrator that directs others. Unlike debate frameworks that use discussion to converge on a shared hypothesis [15, 16], AUTOSCIENTISTS uses discussion to filter out weak proposals before any experiment runs, while allowing agents to continue pursuing different research directions in parallel.

Coordination of Multi-Agent Systems. Beyond scientific applications, multi-agent performance depends on collaboration structure and agent composition [35–40]. Interaction is not automatically beneficial. For example, multi-agent systems have underperformed their best individual member on some tasks [41, 40], and recent benchmarks analyze how collaboration and competition affect collective performance [42]. These findings motivate our ablation studies and comparison to single-agent baselines like Autoresearch. Performance depends on how well the coordination structure matches the task rather than on agent count alone [40], and multi-agent failures often stem from system design, inter-agent misalignment, and weak task verification [43]. In AUTOSCIENTISTS, the shared state, proposal critique, dead-end registry, noise-aware validation, and team reorganization target these limitations. Human scientific teams provide a complementary perspective as they benefit from diversity and flatter structures, but excessive diversity can introduce coordination costs [44–47]. Recent work also emphasizes context management, memory, and reusable skills for sustained collaboration [12, 48, 32]. Our system draws on these findings by organizing agents as teams focused on complementary research directions and using shared forums to support conference style knowledge sharing and collective intelligence [49]. Shared state coordination dates back to blackboard architectures such as Hearsay-II [50, 51]. Recent LLM blackboard systems decentralize which agent contributes next to a shared reasoning thread [52], which agents answer a centrally posted request [53], or which agent claims a predefined work unit [54]. AUTOSCIENTISTS instead decentralizes how research is structured, with agents that create, continue, and reorganize persistent experimental threads and their teams in response to experimental evidence.

3 AUTOSCIENTISTS: Long-Running Self-Organizing Agent Teams

We formalize long-running scientific experimentation as an iterative search process and then introduce AUTOSCIENTISTS. We first define the optimization setting and then describe how agents organize into teams, propose and execute experiments, exchange experimental evidence through a shared state, and reorganize as search trajectories evolve over time.

3.1 Problem Formulation

We are given a task description, optionally accompanied by an initial program (e.g., a training script) p_0, together with a dataset $\mathcal{D}$ and an evaluation metric ℓ.

The dataset $\mathcal{D}$ consists of a training set $\mathcal{D}_{\text{train}}$ and an evaluation protocol. The evaluation protocol may take one of two forms, either a validation set $\mathcal{D}_{\text{val}}$ or a cross-validation (CV) scheme over $\mathcal{D}_{\text{train}}$. We denote by $\ell_{\text{eval}}(p; \mathcal{D})$ the evaluation metric computed under this protocol.

A system of n long-running LLM agents $\mathcal{A} = \{a_1, \ldots, a_n\}$ iteratively proposes and generates new programs. Long-running agents a_i persist over the course of the search process, maintaining internal state and updating their behavior based on accumulated experience. This contrasts with one-shot agents that generate a solution in a single forward pass. Each proposed program p is trained on $\mathcal{D}_{\text{train}}$ and evaluated using ℓ_{eval}. The goal is to identify a program

$$p^* = \arg\max_{p \in \mathcal{P}} \ \ell_{\text{eval}}(p; \mathcal{D}),$$

where $\mathcal{P}$ denotes the space of programs explored by the agents during the search process, optionally initialized from p_0. We assume without loss of generality that ℓ is oriented so that higher values correspond to better performance (e.g., by negating metrics that are typically minimized such as loss).

At the end of the search process, performance is reported using ℓ_{test} if a held-out test set is available, and using ℓ_{eval} (e.g., validation or CV performance) otherwise.

3.2 AUTOSCIENTISTS Approach

Overview. AUTOSCIENTISTS deploys n long-running AI agents that maintain state across the run, adapt their search strategy, reorganize into teams, and update their search behavior from accumulated evidence (Figure 1). The system alternates between two phases. In the *discussion* phase, agents analyze the task, propose experimental directions, and organize into teams. In the *execution* phase, teams run parallel experiments and write results back to the shared state $\mathcal{S}$. When performance on ℓ_{eval} stagnates, agents reopen discussion and may reorganize teams around different directions. This cycle continues for the duration of the run and is coordinated through $\mathcal{S}$ rather than a central planner agent.

Discussion and Self-Organization. Agents identify and revise research directions through discussion phases, without a predefined partition of the search space. AUTOSCIENTISTS initializes with no teams and no predefined directions. At the start of each discussion phase, all agents read the task specification, the current champion p^*, and prior posts on the shared forum $\mathcal{F}$. Discussion proceeds over multiple rounds. Early rounds focus on proposing and evaluating candidate directions, where agents independently analyze p^*, propose modifications, critique competing proposals, and identify gaps in the search space. Later rounds organize agents into K teams $\{\mathcal{T}_1, \ldots, \mathcal{T}_K\}$, where each team is assigned one research direction. The final agent in the discussion round consolidates the proposals into a roster $R = \{(\mathcal{T}_k, \text{direction}_k, \text{members}_k)\}_{k=1}^{K}$ and writes it to $\mathcal{S}$. Subsequent agents adopt the roster on their next heartbeat.

The roster changes as evidence accumulates. When a team stops producing improvements, agents trigger a new discussion phase and review results across all teams. Through the shared research forum, agents can propose to create, merge, split, or rebalance teams, with changes requiring endorsement from affected teams before taking effect. This allows AUTOSCIENTISTS to redirect effort during the run, so exhausted directions can be retired, and newly emerging hypotheses can form new teams.

Long-Running Parallel Experiments. Each team $\mathcal{T}_k$ operates a continuous propose and execute loop. Every agent runs a heartbeat cycle in which it reads the shared state $\mathcal{S}$, acts according to its role, writes results back to $\mathcal{S}$, and repeats. Agents persist across cycles with their own identity and memory files, accumulating knowledge over the duration of the run. Two specialized roles collaborate in each team.

(1) Analyst Agents. Analysts maintain the team's search knowledge and propose experiments. In each heartbeat cycle, an analyst reads the experiment log $\mathcal{L}$, audits which research directions have never been tested, and posts proposals to the team queue Q_k. Proposals are ranked by observed effect sizes from $\mathcal{L}$, where underexplored research directions are prioritized, and research directions with

consistently small effects are deprioritized (details in Appendix A.7). After a champion update, the analyst identifies the features responsible for the improvement and proposes variants that share them.

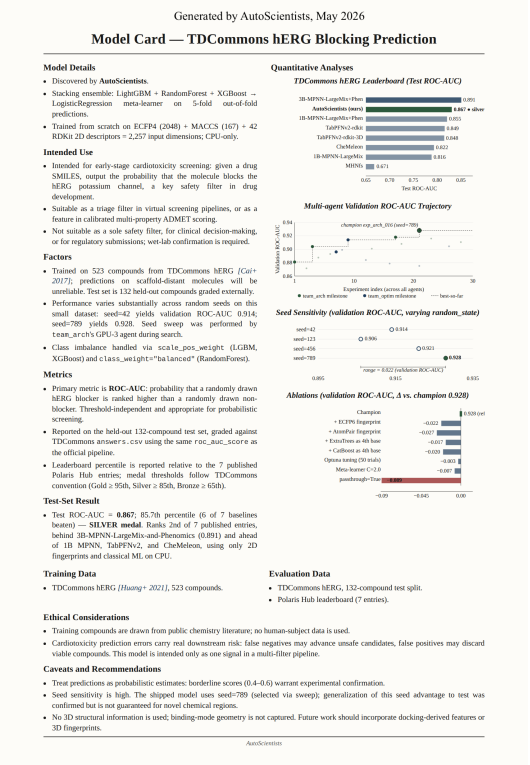
(2) Experiment Agents. Experiment agents claim experiments from the team queue Q_k, apply the code change to p^*, train, and record the outcome in $\mathcal{L}$ and $\mathcal{F}$. Since the evaluation metric ℓ may be stochastic (e.g., variation across training seeds), improvements within the empirically measured noise band are confirmed on a second seed before promotion to p^* (details in Appendix A.6). All results, including failures, are visible to every agent across all teams.

Teams execute in parallel for the full duration. As experiments accumulate, teams track failed experiments in a dead-end registry $\mathcal{D}_k$ to avoid repeating unproductive directions, and rank their queues Q_k by observed effect sizes from $\mathcal{L}$ so that underexplored directions are tried first. When a team's recent experiments consistently fail to improve p^* (e.g., no improvement in the last 10 experiments), agents return to discussion and may reorganize into new teams around more productive directions.

Shared State. The system maintains a shared state accessible to all agents, consisting of a champion p^* tracking the current best model with full hyperparameters and reproduction instructions; an experiment log $\mathcal{L}$ of every completed experiment with outcome, metric delta, and training diagnostics; a shared forum $\mathcal{F}$ of structured posts where proposals are debated, results announced, and mechanistic analyses shared; and team local state (per team experiment queues Q_k, dead-end registries $\mathcal{D}_k$, and hypothesis documents) that is readable cross-team. Details are in Appendix A.

Output. AUTOSCIENTISTS outputs the final champion model p^* together with a model card and a research findings report derived from the agents' experimental process. AUTOSCIENTISTS produces the main technical components of a model card [55]. Model architecture, hyperparameters, and training procedures are recorded in the reproducible champion training script. Training and evaluation datasets are inherited from the shared task specification, and quantitative performance metrics are stored in the champion record. Figure 2 shows a model card for a hERG prediction model discovered by AUTOSCIENTISTS on BioML-Bench.

In addition to the final model, AUTOSCIENTISTS records the experimental search process that produced it. Dead-end registries store failed experimental directions together with the tested research direction, performance change, and rejection reason. Analyst agents document the mechanisms underlying successful modifications and propose related follow-up directions. Combined with the full experiment log, these artifacts provide a record of how hypotheses evolved during the run, which directions were abandoned, and how the final model emerged from accumulated experimental evidence. Appendix E presents the complete set of artifacts produced by AUTOSCIENTISTS on the GPT nanochat task.


Generated by AutoScientists, May 2026

Model Card — TDCommons hERG Blocking Prediction

Model Details
- Discovered by **AutoScientists**.
- Stacking ensemble: LightGBM + RandomForest + XGBoost → LogisticRegression meta-learner on 5-fold out-of-fold predictions.
- Trained from scratch on ECFP4 (2048) + MACCS (167) + 42 RDKit 2D descriptors = 2,257 input dimensions; CPU-only.

Intended Use
- Intended for early-stage cardiotoxicity screening: given a drug SMILES, output the probability that the molecule blocks the hERG potassium channel, a key safety filter in drug development.
- Suitable as a triage filter in virtual screening pipelines, or as a feature in calibrated multi-property ADMET scoring.
- Not suitable as a sole safety filter, for clinical decision-making, or for regulatory submissions; wet-lab confirmation is required.

Factors
- Trained on 523 compounds from TDCommons hERG *[Cai+ 2017]*; predictions on scaffold-distant molecules will be unreliable. Test set is 132 held-out compounds graded externally.
- Performance varies substantially across random seeds on this small dataset: seed=42 yields validation ROC-AUC 0.914; seed=789 yields 0.928. Seed sweep was performed by `team_arch`'s GPU-3 agent during search.
- Class imbalance handled via `scale_pos_weight` (LGBM, XGBoost) and `class_weight="balanced"` (RandomForest).

Metrics
- Primary metric is **ROC-AUC**: probability that a randomly drawn hERG blocker is ranked higher than a randomly drawn non-blocker. Threshold-independent and appropriate for probabilistic screening.
- Reported on the held-out 132-compound test set, graded against TDCommons `answers.csv` using the same `roc_auc_score` as the official pipeline.
- Leaderboard percentile is reported relative to the 7 published Polaris Hub entries; medal thresholds follow TDCommons convention (Gold ≥ 95th, Silver ≥ 85th, Bronze ≥ 65th).

Test-Set Result
- Test ROC-AUC = **0.867**; 85.7th percentile (6 of 7 baselines beaten) — **SILVER medal**. Ranks 2nd of 7 published entries, behind 3B-MPNN-LargeMix-and-Phenomics (0.891) and ahead of 1B MPNN, TabPFNv2, and CheMeleon, using only 2D fingerprints and classical ML on CPU.

Training Data
- TDCommons hERG *[Huang+ 2021]*, 523 compounds.

Evaluation Data
- TDCommons hERG, 132-compound test split.
- Polaris Hub leaderboard (7 entries).

Ethical Considerations
- Training compounds are drawn from public chemistry literature; no human-subject data is used.
- Cardiotoxicity prediction errors carry real downstream risk: false negatives may advance unsafe candidates, false positives may discard viable compounds. This model is intended only as one signal in a multi-filter pipeline.

Caveats and Recommendations
- Treat predictions as probabilistic estimates: borderline scores (0.4–0.6) warrant experimental confirmation.
- Seed sensitivity is high. The shipped model uses seed=789 (selected via sweep); generalization of this seed advantage to test was confirmed but is not guaranteed for novel chemical regions.
- No 3D structural information is used; binding-mode geometry is not captured. Future work should incorporate docking-derived features or 3D fingerprints.

AutoScientists

Figure 2: **Model card produced by AUTOSCIENTISTS.** TDC hERG Blocking Prediction model discovered by AUTOSCIENTISTS.

4 Experiments

4.1 Implementation Details

All agents in AUTOSCIENTISTS use the same base model, the Claude Code coding agent [56] with the base LLM Claude Sonnet 4.6 [57]. We use the same model backend for AUTOSCIENTISTS and the

Autoresearch baseline. To test whether results depend on this backend, we also run AUTOSCIENTISTS with the OpenAI Codex coding agent on two tasks (Appendix C.2). Each agent is repeatedly invoked by a deterministic monitor process in a heartbeat loop. AUTOSCIENTISTS was given access to H100 GPUs for running experiments. Details for reproducing the experimental results are in the Reproducibility Statement. Unless specified otherwise, the AUTOSCIENTISTS team is composed of 3 analyst agents and 6 experiment agents.

4.2 End-to-End Biomedical Machine Learning with AUTOSCIENTISTS

Setup. We evaluate AUTOSCIENTISTS on BioML-Bench, a benchmark of 24 end-to-end biomedical machine learning tasks spanning biomedical imaging (4), drug discovery (9), protein engineering (6), and single-cell omics (5) [2]. Each task provides a natural language task description, training data, test inputs, and an example submission format. For each of the four task types, an LLM generated list of general model paradigms is included in the AUTOSCIENTISTS agent's discussion prompts to encourage diverse research directions. AUTOSCIENTISTS develops models using the task description, training data, and development time validation feedback. Hidden test labels and private grader files are kept outside the agent workspace and are accessed only by the external evaluator, whereas protein engineering tasks are scored with the benchmark's cross-validation protocol (Appendix F). Following BioML-Bench, we report leaderboard percentile relative to public human submissions, whether the submission exceeds the public leaderboard median, whether it receives any medal, and completion rate. We compare against the BioML-Bench baselines Reference, MLAgentBench [58], AIDE [11], STELLA [5], and Biomni [10]. We additionally adapt Autoresearch [3], implemented with the same coding agent backend, to the BioML-Bench tasks. All agents receive the same experimental compute as AUTOSCIENTISTS. Experimental compute settings and task-specific setup details are provided in Appendix F.

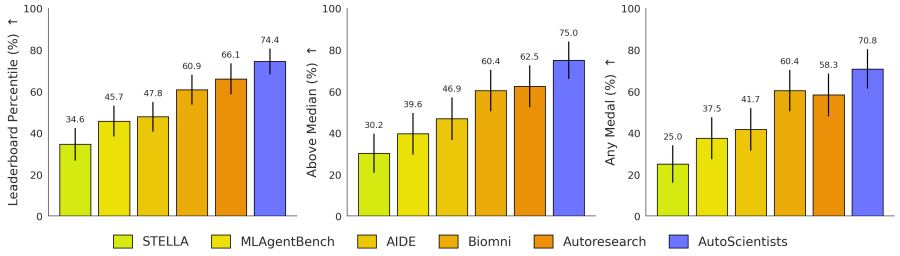


Figure 3: **AUTOSCIENTISTS improves performance across BioML-Bench tasks.** Performance on 24 biomedical tasks measured by leaderboard percentile (**left**), proportion above the public leaderboard median (**middle**), and proportion awarded a medal (**right**). Error bars show standard error of the mean. All agents use the same experimental compute allocation (Appendix F). Additional results are reported in Table S9.

Results. We report aggregate performance in Fig. 3 and domain-level performance in Table 1. Overall, AUTOSCIENTISTS achieves the highest mean (SE) leaderboard percentile among the evaluated systems, with $74.40\,(6.20)\%$ compared with $66.07\,(7.38)\%$ for Autoresearch, an improvement of $+8.33$ leaderboard percentile points. AUTOSCIENTISTS completes all 24 tasks. AUTOSCIENTISTS shows the largest improvement in drug discovery, reaching $64.52\,(8.37)\%$ mean (SE) leaderboard percentile compared with $47.91\,(10.77)\%$ for Biomni. Protein engineering is the strongest domain in absolute leaderboard percentile, but it is largely saturated, with both AUTOSCIENTISTS and Autoresearch obtaining $96.97\,(3.03)\%$, and AUTOSCIENTISTS has a better mean rank of 1.50. Sec. 4.4 instead evaluates whether AUTOSCIENTISTS can discover a single method that transfers across the full ProteinGym supervised substitution benchmark. Biomedical imaging remains the most challenging domain, where each task requires training larger image models. We summarize the final AUTOSCIENTISTS approaches (p^*) in Appendix F.4. To complement the quantitative results, we inspected the shared state and agent logs of AUTOSCIENTISTS to determine whether deliberation changed the experiments selected for execution. Fig. 5 shows representative examples in which agents diversified away from redundant proposals, identified saturated search directions, transferred hypotheses across teams, and retired dead-end directions after stagnation, supporting the interpretation that AUTOSCIENTISTS improves experiment selection under a fixed experiment compute budget.

Table 1: BioML-Bench comparison of models under matched per domain experimental compute budgets, showing Biomni, Autoresearch, and AUTOSCIENTISTS. Values are mean (SE) and rank among all models.

Domain	Biomni		Autoresearch		AUTOSCIENTISTS	
	Leaderboard ↑	Rank ↓	Leaderboard ↑	Rank ↓	Leaderboard ↑	Rank ↓
Biomedical Imaging ($n = 4$)	19.04 (10.83)	4.75	39.60 (21.75)	2.25	**45.75 (22.18)**	**1.50**
Drug Discovery ($n = 9$)	47.91 (10.77)	3.11	46.16 (10.59)	2.89	**64.52 (8.37)**	**2.44**
Protein Engineering ($n = 6$)	93.94 (3.83)	2.50	**96.97 (3.03)**	2.00	**96.97 (3.03)**	**1.50**
Single Cell Omics ($n = 5$)	78.00 (10.20)	3.60	86.00 (9.80)	2.00	**88.00 (9.70)**	**1.60**

Full metrics are reported in Table S9.

4.3 GPT Training Optimization with AUTOSCIENTISTS

We next evaluate whether AUTOSCIENTISTS generalizes beyond biomedical tasks by applying it to GPT nanochat training optimization, the language model training benchmark introduced by Autoresearch [3]. In this task, each experiment modifies a training program and is evaluated by the resulting validation bits-per-byte, so progress depends on selecting useful changes under a fixed experimental budget. This setting tests whether AUTOSCIENTISTS can coordinate search over interacting choices in architecture, optimization, and training schedule.

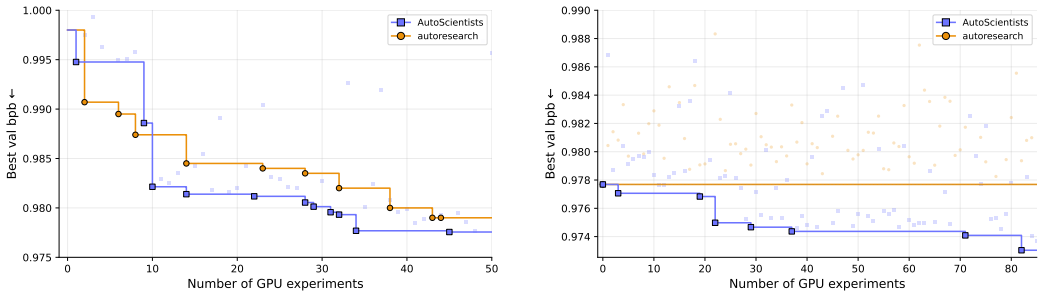
Setup. Each experiment is a single 5-minute GPT training run on one H100 GPU, evaluated by validation bits-per-byte (`val_bpb`, lower is better). We compare AUTOSCIENTISTS against single-agent Autoresearch [3] on the same code repository. We evaluate two regimes, (i) *From Autoresearch baseline*, where both systems start from the same nanochat code at `val_bpb` $= 0.998$ and search for improvements, and (ii) *From an* AUTOSCIENTISTS *champion*, where both systems start from the champion p^* obtained after 50 prior AUTOSCIENTISTS experiments (`val_bpb` $= 0.9777$) and are given identical access to the negative knowledge file `EXPLORED.md` listing previously dead-ended directions. We report per experiment trajectories rather than wall clock so that the comparison isolates orchestration from hardware allocation. Since `val_bpb` on the fixed held-out shard is queried throughout the search, it is a development validation objective. Both regimes are compared in Appendix B.1.

From Autoresearch Baseline. In the from baseline regime (Fig. 4a), AUTOSCIENTISTS reaches `val_bpb` ≈ 0.978 in 34 experiments, while single-agent Autoresearch reaches the same value only after 65 experiments. From experiment 10 onward, AUTOSCIENTISTS stays lower for the remainder of the comparison window. The improvement comes from parallel exploration by self-organized teams. Agents in this run formed three teams (architecture, schedule, and optimizer) that propose and run experiments concurrently, so multiple research directions advance within a single agent cycle, whereas the single-agent loop advances one research direction per experiment.

From an AUTOSCIENTISTS Champion. In the from champion regime (Fig. 4b), both methods start from the same AUTOSCIENTISTS champion at `val_bpb` $= 0.9777$. AUTOSCIENTISTS accepts seven improvements over 93 experiments and reaches `val_bpb` $= 0.9730$. Single-agent Autoresearch accepts *zero* improvements over 100 experiments, and its best attempt reaches `val_bpb` $= 0.9783$. The seven AUTOSCIENTISTS improvements span heterogeneous research directions (query-key normalization order, matrix initialization, value embedding gate width, final learning rate fraction, softcap value, compile autotuning, and a noise-floor recalibration of the starting champion) rather than concentrating on a single direction. The first improvement AUTOSCIENTISTS discovered, query-key normalization order, was never proposed by the single-agent loop in any of its 100 attempts, indicating that the improvement comes from a wider set of hypotheses rather than from more compute. In contrast, the single-agent loop repeatedly perturbed settings that were already near local optima in the starting champion's tuned configuration and produced only null results.

4.4 Extending Kermut on ProteinGym Supervised Fitness Prediction

Here we evaluate whether AUTOSCIENTISTS can improve a strong existing scientific codebase rather than solve a benchmark from scratch. This setting reflects how scientific research is typically conducted, where researchers begin from the strongest available method and iteratively search for modifications that improve it.



(a) From Autoresearch baseline

(b) From an AUTOSCIENTISTS champion

Figure 4: **AUTOSCIENTISTS sustains improvement during long-running GPT training optimization.** GPT nanochat training optimization with AUTOSCIENTISTS and Autoresearch [3]. **(a) From Autoresearch baseline** (`val_bpb` $= 0.998$), AUTOSCIENTISTS reaches `val_bpb` ≈ 0.978 in 34 experiments vs. 65 for Autoresearch, a $1.9\times$ speedup at the matched loss. **(b) From an AUTOSCIENTISTS champion** obtained after 50 prior AUTOSCIENTISTS experiments (`val_bpb` $= 0.9777$) with the same dead-end registry, AUTOSCIENTISTS accepts 7 out of 93 experiments to reach `val_bpb` $= 0.9730$, whereas Autoresearch accepts 0 out of 100.

Setup. We use Kermut [59], a Gaussian process method for supervised protein variant effect prediction, as the seed program for AUTOSCIENTISTS because it is the best-performing baseline. We refer to the resulting discovered model as AUTOSCIENTISTS-Kermut. During development, AUTOSCIENTISTS is allowed to modify the Kermut implementation in any way and is evaluated on a single development assay, ACE2–Spike binding (ProteinGym assay `SPIKE_SARS2_Starr_2020_binding`, hereafter SPIKE-SARS2). Agents optimize the objective $\ell_{\text{eval}} = \frac{1}{3}\sum_s \rho_s$, where $s \in \{\text{random}, \text{modulo}, \text{contiguous}\}$ and ρ_s is the out-of-fold Spearman correlation under split scheme s. We choose this assay because Kermut obtains a low mean Spearman's ρ, making it a challenging development target. AUTOSCIENTISTS is run for 10 cycles with no human intervention and access to one H100 GPU. One cycle corresponds to each experiment agent completing one queued experiment. After development, the discovered recipe is frozen and applied without further modification to the full ProteinGym supervised substitution benchmark of 217 DMS assays.

Results. On the ACE2–Spike development assay, across three independent AUTOSCIENTISTS runs from the Kermut model, AUTOSCIENTISTS-Kermut reaches 0.838 ± 0.004, versus 0.809 ± 0.006 for three single-agent Autoresearch runs from the same Kermut model and number of experiments (Appendix C.2). This Kermut seeded setting differs from the cold start BioML-Bench SPIKE-SARS2 task in Table S4, so the two results are not comparable. We analyze the best performing AUTOSCIENTISTS-Kermut champion, where the discovered predictor is a three-GP ensemble that combines Kermut's structure kernel with expanded zero-shot features, greedy diversity-based feature selection, and quantile warped targets. AUTOSCIENTISTS explores research directions beyond hyperparameter tuning of Kermut. Architectural details are provided in Appendix G with ablations of AUTOSCIENTISTS-introduced components in Appendix G.2. We then evaluate the frozen AUTOSCIENTISTS-Kermut recipe across all 217 supervised substitution DMS assays in ProteinGym. As shown in Table 2, AUTOSCIENTISTS-Kermut improves the official average Spearman's ρ from 0.657 for Kermut to 0.700, an absolute improvement of 0.043 and a 6.5% relative improvement. The improvement in Spearman's ρ is observed across all three CV schemes. AUTOSCIENTISTS-Kermut also outperforms the other supervised baselines considered. The discovered quantile warping and rank oriented model selection improve variant ordering but not calibrated regression, as MSE increases by 0.006. Extending AUTOSCIENTISTS to optimize multi-objective leaderboards, including MSE, is an important direction for future work.

4.5 Ablations of AUTOSCIENTISTS

We next test which components of AUTOSCIENTISTS are responsible for its performance. The ablations remove one component at a time while keeping the agent backend, task interface, starting program, and experimental budget fixed. This tests whether improvements come from analyst guided proposal generation, cross-agent feedback, team reorganization, or the shared experimental record.

Setup. We identify the contribution of four AUTOSCIENTISTS components on four tasks, TDC-hERG, Cell-Cell Communication, Human Plasma-Protein Binding, and GPT nanochat training optimization

Table 2: Performance on the ProteinGym supervised substitution benchmark, comprising 217 DMS assays across UniProt/function groups. Best results are in **bold** and second best in *italic*. Values are mean (SE).

Model type	Model name	Spearman's ρ (↑)				MSE (↓)			
		Contig.	Mod.	Rand.	Avg.	Contig.	Mod.	Rand.	Avg.
Embed.	ESM-1v Embeddings	0.479 (0.015)	0.514 (0.014)	0.614 (0.019)	0.535 (0.009)	0.914 (0.053)	0.848 (0.071)	0.603 (0.055)	0.789 (0.035)
	MSA Transformer Embeddings	0.513 (0.019)	0.562 (0.016)	0.670 (0.012)	0.581 (0.009)	0.860 (0.055)	0.783 (0.066)	0.529 (0.030)	0.724 (0.030)
	Tranception Embeddings	0.439 (0.014)	0.525 (0.011)	0.681 (0.015)	0.548 (0.008)	1.037 (0.048)	0.849 (0.052)	0.518 (0.043)	0.802 (0.028)
NPT	ProteinNPT	0.529 (0.018)	0.588 (0.013)	0.741 (0.015)	0.619 (0.009)	0.856 (0.051)	0.765 (0.056)	0.441 (0.046)	0.687 (0.029)
Kermut	Kermut	*0.593 (0.015)*	*0.633 (0.012)*	*0.746 (0.014)*	*0.657 (0.008)*	**0.737 (0.042)**	*0.666 (0.050)*	**0.413 (0.035)**	**0.605 (0.024)**
	AUTOSCIENTISTS	**0.635 (0.013)**	**0.681 (0.011)**	**0.783 (0.010)**	**0.700 (0.007)**	*0.812 (0.131)*	**0.607 (0.040)**	**0.413 (0.025)**	*0.611 (0.047)*

(Table 3), holding the agent backend, task interface, total compute, and starting program fixed. The four ablations are (1) **No analyst** removes the three analyst agents and reassigns their duties (proposal generation, knowledge file maintenance, hypothesis tracking) to the experiment agents; (2) **No cross-agent feedback** disables comment threads on proposals and results, so agents cannot critique each other or share near-misses across teams; (3) **No self-organization** fixes a predefined team organization at the start of the run and prevents agents from reorganizing teams across rounds, while keeping parallel agents, the shared state, and cross-agent feedback; (4) **Independent agents** removes both cross-agent feedback and the shared state (champion program, results log, dead-end registry, and accumulated knowledge), so each agent runs a solo loop maintaining only its own private state and cannot observe any other agent's results.

Results. The full AUTOSCIENTISTS wins every task on its primary metric, and the most damaging ablation differs by task. Removing the *analyst* is most damaging on TDC-hERG (AUROC $0.867 \rightarrow 0.738$, leaderboard percentile $85.7 \rightarrow 14.3$). Removing *cross-agent feedback* is most damaging on Human Plasma-Protein Binding (Pearson correlation $0.8729 \rightarrow 0.7144$, leaderboard percentile $80 \rightarrow 30$). Removing *self-organization* is most damaging on GPT training optimization (`val_bpb` $0.9777 \rightarrow 0.9833$). The *independent agents* ablation is most damaging on Cell-Cell Communication (Odds Ratio $0.924 \rightarrow 0.435$, the largest proportional drop in the table). No single mechanism dominates across all four tasks and each removed component produces an AUTOSCIENTISTS that is dominated on at least one task. We read this as evidence that the four components address complementary failure modes rather than redundantly contributing the same kind of improvement. Analyst driven refinement matters when proposal quality is the bottleneck, cross-agent feedback matters when individual agents observe only a partial signal, self-organization matters when the productive search direction shifts during the run, and a shared experimental record matters when isolated agents would otherwise duplicate work or converge to incompatible local optima. In addition to removing *self-organization*, we further explore how team organization affects AUTOSCIENTISTS performance on GPT training optimization in Appendix B.2. Letting agents form initial teams that then stay fixed reaches a `val_bpb` of 0.9806, and starting from the same initial teams and hypotheses that AUTOSCIENTISTS forms reaches 0.9794, compared with 0.9777 for AUTOSCIENTISTS. Per ablation trajectories are in Appendix D.

Table 3: Ablation results of AUTOSCIENTISTS for biomedicine and GPT training optimization tasks.

Task	Metric	No analyst	No cross-agent	No self-org.	Independent agents	AUTOSCIENTISTS
TDC-hERG	AUROC (↑)	0.738	0.819	0.807	0.853	**0.867**
Cell-Cell Communication	Odds Ratio (↑)	0.858	0.908	0.628	0.435	**0.924**
Human Plasma-Protein Binding	Pearson correlation (↑)	0.813	0.714	0.811	0.784	**0.873**
GPT Training Optimization	Best `val_bpb` (↓)	0.9817	0.9814	0.9833	0.9833	**0.9777**

5 Limitations and Future Work

AUTOSCIENTISTS is not designed to be more LLM call efficient than single-agent baselines. As shown in Table S11, AUTOSCIENTISTS uses more LLM tokens than Autoresearch, though within the same order of magnitude, reflecting its use of multiple agents for parallel reasoning, discussion, and team reorganization. Instead, AUTOSCIENTISTS is designed to improve experimental search

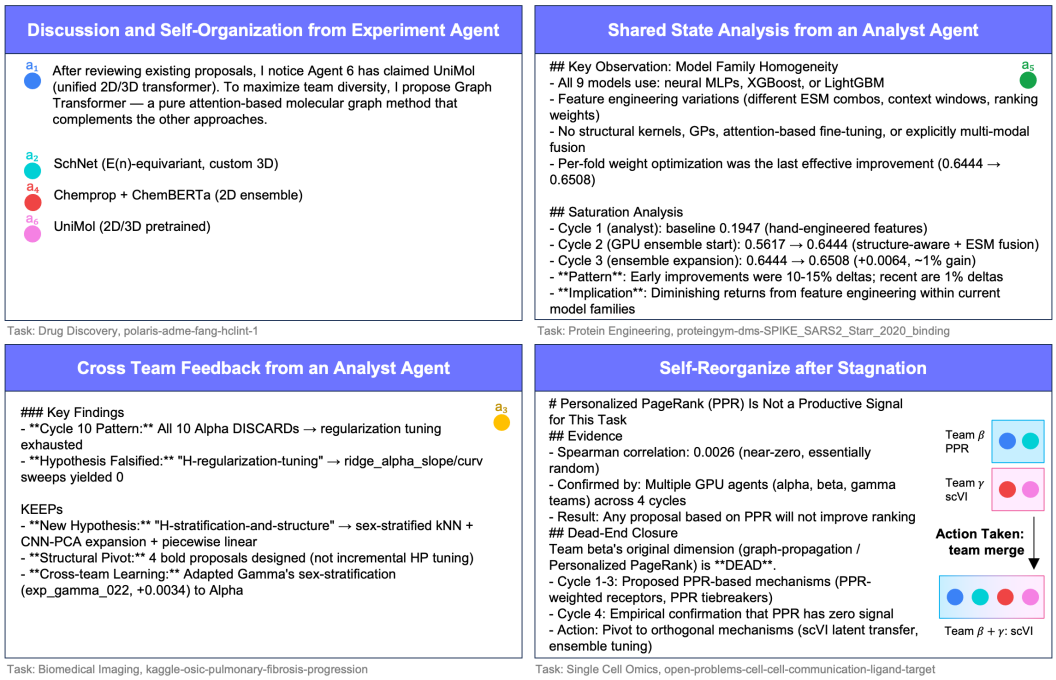


Figure 5: **Emergent coordination during long-running experimental search.** Illustrations of AUTOSCIENTISTS agent team interactions in long-running research experiments, featuring representative quotes from the agents.

under a fixed experimental compute budget by enabling teams of agents to explore and collaborate over the design space. Our comparisons match experimental compute but not LLM token usage, and the higher token usage of AUTOSCIENTISTS may limit accessibility. Token usage could be reduced with LLM routing [60], deactivating low contributing agents [61], or pruning messages that do not affect decisions [62]. Although AUTOSCIENTISTS uses Claude Code in the main experiments, runs with an OpenAI Codex backend reach consistent results (Appendix C.2). As part of the matched experimental compute budget used for fair comparison on BioML-Bench, we restricted AUTOSCIENTISTS to one H100 GPU per task so GPU-bound experiments in that evaluation were executed sequentially. This setting evaluates experiment selection under matched compute but does not fully exercise AUTOSCIENTISTS's capacity for parallel experimentation. When multiple GPUs are available, AUTOSCIENTISTS can dispatch experiments concurrently. With 2 and 4 GPUs, wall-clock time per experiment on GPT nanochat drops by $1.52\times$ and $2.18\times$, and the speedup is sublinear because the LLM reasoning in each experiment does not shrink (Appendix B.3). Evaluating how AUTOSCIENTISTS scales with larger GPU budgets and asynchronous dispatch remains future work. Additionally, the number of agents is set before the run, and nine agents is not optimal for every task. In future work, we will scale team size dynamically with task difficulty. We provide preliminary exploration of different AUTOSCIENTISTS team sizes and practical guidance in Appendix B.3.

6 Conclusion

We introduced AUTOSCIENTISTS, a self-organizing agent team for long-horizon scientific experimentation. AUTOSCIENTISTS addresses a limitation of existing AI agents, which can run individual experiments but often struggle to sustain experimental search as evidence accumulates and productive directions change. Across BioML-Bench, GPT training optimization, and ProteinGym, AUTOSCIENTISTS improves over state-of-the-art AI agents under matched experimental budgets.

AUTOSCIENTISTS makes long-running experimentation a collective search process. Agents evaluate proposals before execution, record successful and failed directions, share evidence through a common state, and reorganize teams when progress stalls. This design helps agents use experimental trials more effectively, avoid repeated dead ends, and continue identifying productive modifications after individual agents plateau.

Acknowledgments

A.F. is supported by the Kempner Graduate Fellowship at Harvard University. We gratefully acknowledge the support by NSF CAREER Award 2339524, ARPA-H Biomedical Data Fabric (BDF) Toolbox Program, Amazon Faculty Research, Google Research Scholar Program, AstraZeneca Research, GlaxoSmithKline Award, Roche Alliance with Distinguished Scientists (ROADS) Program, Sanofi iDEA-iTECH Award, Boehringer Ingelheim Award, Merck Award, Optum AI Research Collaboration Award, Pfizer Research, Gates Foundation (INV-079038), Chan Zuckerberg Initiative, Collaborative Center for XDP at Massachusetts General Hospital, John and Virginia Kaneb Fellowship at Harvard Medical School, Biswas Computational Biology Initiative in partnership with the Milken Institute, Harvard Medical School Dean's Innovation Fund for the Use of Artificial Intelligence, and the Kempner Institute for the Study of Natural and Artificial Intelligence at Harvard University. This work was delivered as part of the AURORA project supported by the Cancer Grand Challenges partnership funded by Cancer Research UK ([CGCAI1-Mar26/100001]). Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funders.

References

[1] Shanghua Gao, Ada Fang, Yepeng Huang, Valentina Giunchiglia, Ayush Noori, Jonathan Richard Schwarz, Yasha Ektefaie, Jovana Kondic, and Marinka Zitnik. Empowering biomedical discovery with ai agents. *Cell*, 187(22):6125–6151, 2024.

[2] Henry E. Miller, Matthew Greenig, Benjamin Tenmann, and Bo Wang. BioML-bench: Evaluation of AI agents for end-to-end biomedical ML. *bioRxiv*, 2025. doi: 10.1101/2025.09.01.673319. URL https://www.biorxiv.org/content/early/2025/09/28/2025.09.01.673319.

[3] Andrej Karpathy. Autoresearch: AI agents running research on single-GPU nanochat training automatically. https://github.com/karpathy/autoresearch, 2026. GitHub repository.

[4] Ludovico Mitchener, Angela Yiu, Benjamin Chang, Mathieu Bourdenx, Tyler Nadolski, Arvis Sulovari, Eric C Landsness, Daniel L Barabasi, Siddharth Narayanan, Nicky Evans, et al. Kosmos: An AI scientist for autonomous discovery. *arXiv preprint arXiv:2511.02824*, 2025.

[5] Ruofan Jin, Mingyang Xu, Fei Meng, Guancheng Wan, Qingran Cai, Yize Jiang, Jin Han, Yuanyuan Chen, Wanqing Lu, Mengyang Wang, Zhiqian Lan, Yuxuan Jiang, Junhong Liu, Dongyao Wang, Le Cong, and Zaixi Zhang. Stella: Towards a biomedical world model with self-evolving multimodal agents. *bioRxiv*, 2026. doi: 10.1101/2025.07.01.662467. URL https://www.biorxiv.org/content/early/2026/01/25/2025.07.01.662467.

[6] Shanghua Gao, Richard Zhu, Zhenglun Kong, Ayush Noori, Xiaorui Su, Curtis Ginder, Theodoros Tsiligkaridis, and Marinka Zitnik. Txagent: an ai agent for therapeutic reasoning across a universe of tools. *arXiv preprint arXiv:2503.10970*, 2025.

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

[8] José R Penadés, Juraj Gottweis, Lingchen He, Jonasz B Patkowski, Alexander Daryin, Wei-Hung Weng, Tao Tu, Anil Palepu, Artiom Myaskovsky, Annalisa Pawlosky, et al. Ai mirrors experimental science to uncover a mechanism of gene transfer crucial to bacterial evolution. *Cell*, 188(23):6654–6665, 2025.

[9] Pengwei Sui, Michelle M. Li, Shanghua Gao, Wanxiang Shen, Valentina Giunchiglia, Andrew Shen, Yepeng Huang, Zhenglun Kong, and Marinka Zitnik. Medea: An omics ai agent for therapeutic discovery. *bioRxiv*, 2026. doi: 10.64898/2026.01.16.696667. URL https://www.biorxiv.org/content/early/2026/01/20/2026.01.16.696667.

[10] Kexin Huang, Serena Zhang, Hanchen Wang, Yuanhao Qu, Yingzhou Lu, Yusuf Roohani, Ryan Li, Lin Qiu, Gavin Li, Junze Zhang, et al. Biomni: A general-purpose biomedical AI agent. *biorxiv*, 2025.

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

[12] Ao Qu, Han Zheng, Zijian Zhou, Yihao Yan, Yihong Tang, Shao Yong Ong, Fenglu Hong, Kaichen Zhou, Chonghe Jiang, Minwei Kong, et al. Coral: Towards autonomous multi-agent evolution for open-ended discovery. *arXiv preprint arXiv:2604.01658*, 2026.

[13] Shiyang Feng, Runmin Ma, Xiangchao Yan, Yue Fan, Yusong Hu, Songtao Huang, Shuaiyu Zhang, Zongsheng Cao, Tianshuo Peng, Jiakang Yuan, et al. Internagent-1.5: A unified agentic framework for long-horizon autonomous scientific discovery. *arXiv preprint arXiv:2602.08990*, 2026.

[14] Kyle Swanson, Wesley Wu, Nash L Bulaong, John E Pak, and James Zou. The virtual lab of ai agents designs new sars-cov-2 nanobodies. *Nature*, 646(8085):716–723, 2025.

[15] Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first international conference on machine learning*, 2024.

[16] Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. ReConcile: Round-table conference improves reasoning via consensus among diverse LLMs, 2024. URL https://arxiv.org/abs/2309.13007.

[17] Pascal Notin, Aaron Kollasch, Daniel Ritter, Lood van Niekerk, Steffanie Paul, Han Spinner, Nathan Rollins, Ada Shaw, Rose Orenbuch, Ruben Weitzman, Jonathan Frazer, Mafalda Dias, Dinko Franceschi, Yarin Gal, and Debora Marks. Proteingym: Large-scale benchmarks for protein fitness prediction and design. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 64331–64379. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/cac723e5ff29f65e3fcbb0739ae91bee-Paper-Datasets_and_Benchmarks.pdf.

[18] Mourad Gridach, Jay Nanavati, Khaldoun Zine El Abidine, Lenon Mendes, and Christina Mack. Agentic AI for scientific discovery: A survey of progress, challenges, and future directions, 2025. URL https://arxiv.org/abs/2503.08979.

[19] Chengwei Liu, Chong Wang, Jiayue Cao, Jingquan Ge, Kun Wang, Lyuye Zhang, Ming-Ming Cheng, Penghai Zhao, Tianlin Li, Xiaojun Jia, Xiang Li, Xingshuai Li, Yang Liu, Yebo Feng, Yihao Huang, Yijia Xu, Yuqiang Sun, Zhenhong Zhou, and Zhengzi Xu. A vision for auto research with LLM agents, 2025. URL https://arxiv.org/abs/2504.18765.

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

[21] Jonathan Bragg, Mike D'Arcy, Nishant Balepur, Dan Bareket, Bhavana Dalvi, Sergey Feldman, Dany Haddad, Jena D. Hwang, Peter Jansen, Varsha Kishore, Bodhisattwa Prasad Majumder, Aakanksha Naik, Sigal Rahamimov, Kyle Richardson, Amanpreet Singh, Harshit Surana, Aryeh Tiktinsky, Rosni Vasu, Guy Wiener, Chloe Anastasiades, Stefan Candra, Jason Dunkelberger, Dan Emery, Rob Evans, Malachi Hamada, Regan Huff, Rodney Kinney, Matt Latzke, Jaron Lochner, Ruben Lozano-Aguilera, Cecile Nguyen, Smita Rao, Amber Tanaka, Brooke Vlahos, Peter Clark, Doug Downey, Yoav Goldberg, Ashish Sabharwal, and Daniel S. Weld. AstaBench: Rigorous benchmarking of AI agents with a scientific research suite, 2025. URL https://arxiv.org/abs/2510.21652.

[22] 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.

[23] Dechao Bu, Jingbo Sun, Kun Li, Zihao He, Wei Huang, Jinlin Hu, Shanshan Zhang, Shuangshuang Lei, Peipei Huo, Zhihao Wang, et al. Empowering ai data scientists using a multi-agent llm framework with self-evolving capabilities for autonomous, tool-aware biomedical data analyses. *Nature Biomedical Engineering*, pages 1–16, 2026.

[24] Ali Essam Ghareeb, Benjamin Chang, Ludovico Mitchener, Angela Yiu, Caralyn J Szostkiewicz, Jon M Laurent, Muhammed T Razzak, Andrew D White, Michaela M Hinks, and Samuel G Rodriques. Robin: A multi-agent system for automating scientific discovery. *arXiv preprint arXiv:2505.13400*, 2025.

[25] Haoyang Liu, Yijiang Li, and Haohan Wang. GenoMAS: A multi-agent framework for scientific discovery via code-driven gene expression analysis. *arXiv preprint arXiv:2507.21035*, 2025.

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

[27] Keyan Ding, Jing Yu, Junjie Huang, Yuchen Yang, Qiang Zhang, and Huajun Chen. Scitoolagent: a knowledge-graph-driven scientific agent for multitool integration. *Nature Computational Science*, 5(10):962–972, 2025.

[28] Dong Han, Zhehong Ai, Pengxiang Cai, Shanya Lu, Jianpeng Chen, Zihao Ye, Shuzhou Sun, Ben Gao, Lingli Ge, Weida Wang, et al. ChemBOMAS: Accelerated BO in chemistry with LLM-enhanced multi-agent system. *arXiv preprint arXiv:2509.08736*, 2025.

[29] Shijie Xia, Yuhan Sun, and Pengfei Liu. SR-scientist: Scientific equation discovery with agentic AI. *arXiv preprint arXiv:2510.11661*, 2025.

[30] Xiao Wu, Ting-Zhu Huang, Liang-Jian Deng, Xiaobing Yu, Yu Zhong, Shangqi Deng, Ufaq Khan, Jianghao Wu, Xiaofeng Liu, Imran Razzak, Xiaojun Chang, and Yutong Xie. SelfAI: A self-directed framework for long-horizon scientific discovery, 2025. URL https://arxiv.org/abs/2512.00403.

[31] 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, 2026. URL https://arxiv.org/abs/2603.08127.

[32] Xu Huang, Junwu Chen, Yuxing Fei, Zhuohan Li, Philippe Schwaller, and Gerbrand Ceder. CASCADE: Cumulative agentic skill creation through autonomous development and evolution. *arXiv preprint arXiv:2512.23880*, 2025.

[33] 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.

[34] Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

[35] Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Language agents as optimizable graphs. *arXiv preprint arXiv:2402.16823*, 2024.

[36] Jen tse Huang, Jiaxu Zhou, Tailin Jin, Xuhui Zhou, Zixi Chen, Wenxuan Wang, Youliang Yuan, Michael Lyu, and Maarten Sap. On the resilience of LLM-based multi-agent collaboration with faulty agents. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=bkiM54QftZ.

[37] Frédéric Berdoz, Leonardo Rugli, and Roger Wattenhofer. Can ai agents agree? *arXiv preprint arXiv:2603.01213*, 2026.

[38] Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D. Nguyen. Multi-agent collaboration mechanisms: A survey of LLMs, 2025. URL https://arxiv.org/abs/2501.06322.

[39] Yingxuan Yang, Chengrui Qu, Muning Wen, Laixi Shi, Ying Wen, Weinan Zhang, Adam Wierman, and Shangding Gu. Understanding agent scaling in LLM-based multi-agent systems via diversity. *arXiv preprint arXiv:2602.03794*, 2026.

[40] Yubin Kim, Ken Gu, Chanwoo Park, Chunjong Park, Samuel Schmidgall, A Ali Heydari, Yao Yan, Zhihan Zhang, Yuchen Zhuang, Mark Malhotra, et al. Towards a science of scaling agent systems. *arXiv preprint arXiv:2512.08296*, 2025.

[41] Aneesh Pappu, Batu El, Hancheng Cao, Carmelo di Nolfo, Yanchao Sun, Meng Cao, and James Zou. Multi-agent teams hold experts back. *arXiv preprint arXiv:2602.01011*, 2026.

[42] Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, and Jiaxuan You. MultiAgentBench: Evaluating the collaboration and competition of LLM agents, 2025. URL https://arxiv.org/abs/2503.01935.

[43] Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent LLM systems fail? *arXiv preprint arXiv:2503.13657*, 2025.

[44] Jonathon N Cummings and Sara Kiesler. Collaborative research across disciplinary and organizational boundaries. *Social studies of science*, 35(5):703–722, 2005.

[45] Stefan Wuchty, Benjamin F Jones, and Brian Uzzi. The increasing dominance of teams in production of knowledge. *Science*, 316(5827):1036–1039, 2007.

[46] Fengli Xu, Lingfei Wu, and James Evans. Flat teams drive scientific innovation. *Proceedings of the National Academy of Sciences*, 119(23):e2200927119, 2022.

[47] Kara L Hall, Amanda L Vogel, Grace C Huang, Katrina J Serrano, Elise L Rice, Sophia P Tsakraklides, and Stephen M Fiore. The science of team science: A review of the empirical evidence and research gaps on collaboration in science. *American psychologist*, 73(4):532, 2018.

[48] Xinyu Zhu, Yuzhu Cai, Zexi Liu, Bingyang Zheng, Cheng Wang, Rui Ye, Jiaao Chen, Hanrui Wang, Wei-Chen Wang, Yuzhi Zhang, et al. Toward ultra-long-horizon agentic science: Cognitive accumulation for machine learning engineering. *arXiv preprint arXiv:2601.10402*, 2026.

[49] Christoph Riedl. Emergent coordination in multi-agent language models. *arXiv preprint arXiv:2510.05174*, 2025.

[50] Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy. The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. *ACM Computing Surveys*, 12(2):213–253, 1980.

[51] H. Penny Nii. Blackboard systems: The blackboard model of problem solving and the evolution of blackboard architectures. *AI Magazine*, 7(2):38–53, 1986.

[52] Bochen Han and Songmao Zhang. Exploring advanced LLM multi-agent systems based on blackboard architecture. *arXiv preprint arXiv:2507.01701*, 2025.

[53] Alireza Salemi, Mihir Parmar, Palash Goyal, Yiwen Song, Jinsung Yoon, Hamed Zamani, Tomas Pfister, and Hamid Palangi. LLM-based multi-agent blackboard system for information discovery in data science. *arXiv preprint arXiv:2510.01285*, 2025.

[54] Sergey Pugachev. CodeCRDT: Observation-driven coordination for multi-agent LLM code generation. *arXiv preprint arXiv:2510.18893*, 2025.

[55] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 220–229, 2019.

[56] Anthropic. Claude Code: Overview. https://code.claude.com/docs/en/overview, 2026. Product documentation. Accessed: 2026-05-06.

[57] Anthropic. Claude Sonnet 4.6. https://www.anthropic.com/claude/sonnet, 2026. Model documentation. Model ID: `claude-sonnet-4-6`. Accessed: 2026-05-06.

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

[59] Peter Mø rch Groth, Mads Herbert Kerrn, Lars Olsen, Jesper Salomon, and Wouter Boomsma. Kermut: Composite kernel regression for protein variant effects. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 29514–29565. Curran Associates, Inc., 2024. doi: 10.52202/079017-0929. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/34547650b2ca69d91f3b3c3ae8b21962-Paper-Conference.pdf.

[60] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs with preference data. In *International Conference on Learning Representations*, 2025.

[61] Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic LLM-powered agent network for task-oriented agent collaboration. In *Conference on Language Modeling*, 2024.

[62] Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for LLM-based multi-agent systems. In *International Conference on Learning Representations*, 2025.

[63] OpenAI. Codex. https://github.com/openai/codex, 2026. Coding agent. Accessed: 2026-08-01.

[64] S. L. Lee, P. Yadav, Y. Li, J. J. Meudt, J. Strang, D. Hebel, A. Alfson, S. J. Olson, T. R. Kruser, J. B. Smilowitz, K. Borchert, B. Loritz, L. Gharzai, S. Karimpour, J. Bayouth, and M. F. Bassetti. Uw-madison gi tract image segmentation. https://kaggle.com/competitions/uw-madison-gi-tract-image-segmentation, 2022. Kaggle.

[65] Ahmed Shahin, Carmela Wegworth, David, Elizabeth Estes, Julia Elliott, Justin Zita, SimonWalsh, Slepetys, and Will Cukierski. Osic pulmonary fibrosis progression. https://kaggle.com/competitions/osic-pulmonary-fibrosis-progression, 2020. Kaggle.

[66] Will Cukierski. Histopathologic cancer detection. https://kaggle.com/competitions/histopathologic-cancer-detection, 2018. Kaggle.

[67] Adam Flanders, Chris Carr, Evan Calabrese, PhD FelipeKitamura, MD, inversion, JeffRudie, John Mongan, Julia Elliott, Luciano Prevedello, Michelle Riopel, sprint, Spyridon Bakas, and Ujjwal. Rsna-miccai brain tumor radiogenomic classification. https://kaggle.com/competitions/rsna-miccai-brain-tumor-radiogenomic-classification, 2021. Kaggle.

[68] Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohani, Jure Leskovec, Connor W Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. *arXiv preprint arXiv:2102.09548*, 2021.

[69] Polaris. Polaris: The benchmarking platform for drug discovery. https://polarishub.io/, 2026. Accessed: May 2026.

[70] Malte D Luecken, Scott Gigante, Daniel B Burkhardt, Robrecht Cannoodt, Daniel C Strobl, Nikolay S Markov, Luke Zappia, Giovanni Palla, Wesley Lewis, Daniel Dimitrov, et al. Defining and benchmarking open problems in single-cell analysis. *Nature Biotechnology*, 43(7):1035–1040, 2025.

[71] 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.

[72] Justas Dauparas, Ivan Anishchenko, Nathaniel Bennett, Hua Bai, Robert J Ragotte, Lukas F Milles, Basile IM Wicky, Alexis Courbet, Rob J de Haas, Neville Bethel, et al. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56, 2022.

[73] Seyone Chithrananda, Gabriel Grand, and Bharath Ramsundar. Chemberta: large-scale self-supervised pretraining for molecular property prediction. *arXiv preprint arXiv:2010.09885*, 2020.

[74] Esther Heid, Kevin P Greenman, Yunsie Chung, Shih-Cheng Li, David E Graff, Florence H Vermeire, Haoyang Wu, William H Green, and Charles J McGill. Chemprop: a machine learning package for chemical property prediction. *Journal of chemical information and modeling*, 64 (1):9–17, 2024.

[75] RDKit: Open-source cheminformatics. https://www.rdkit.org, 2026. Accessed: May 2026.

[76] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785. URL http://doi.acm.org/10.1145/2939672.2939785.

[77] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[78] Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. Catboost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*, 2018.

[79] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[80] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[81] Kevin K Yang, Niccolò Zanichelli, and Hugh Yeh. Masked inverse folding with sequence transfer for protein representation learning. *Protein Engineering, Design and Selection*, 36: gzad015, 2023.

[82] Yang Tan, Ruilin Wang, Banghao Wu, Liang Hong, and Bingxin Zhou. From high-throughput evaluation to wet-lab studies: advancing mutation effect prediction with a retrieval-enhanced model. *Bioinformatics*, 41(Supplement 1):i401–i409, 07 2025. doi: 10.1093/bioinformatics/btaf189. URL https://doi.org/10.1093/bioinformatics/btaf189.

[83] Mingchen Li, Yang Tan, Xinzhu Ma, Bozitao Zhong, Huiqun Yu, Ziyi Zhou, Wanli Ouyang, Bingxin Zhou, Pan Tan, and Liang Hong. Prosst: Protein language modeling with quantized structure and disentangled attention. *Advances in Neural Information Processing Systems*, 37: 35700–35726, 2024.

[84] Matsvei Tsishyn, Pauline Hermans, Marianne Rooman, and Fabrizio Pucci. Residue conservation and solvent accessibility are (almost) all you need for predicting mutational effects in proteins. *Bioinformatics*, 41(6):btaf322, 2025.

[85] Mustafa Tekpinar, Laurent David, Thomas Henry, and Alessandra Carbone. Prescott: a population aware, epistatic, and structural model accurately predicts missense effects. *Genome Biology*, 26(1):113, 2025.

[86] Bo Chen, Xingyi Cheng, Pan Li, Yangli-ao Geng, Jing Gong, Shen Li, Zhilei Bei, Xu Tan, Boyan Wang, Xin Zeng, et al. xtrimopglm: unified 100-billion-parameter pretrained transformer for deciphering the language of proteins. *Nature Methods*, 22(5):1028–1039, 2025.

[87] Jin Su, Chenchen Han, Yuyang Zhou, Junjie Shan, Xibin Zhou, and Fajie Yuan. Saprot: Protein language modeling with structure-aware vocabulary. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=6MRm3G4NiU.

[88] Chloe Hsu, Robert Verkuil, Jason Liu, Zeming Lin, Brian Hie, Tom Sercu, Adam Lerer, and Alexander Rives. Learning inverse folding from millions of predicted structures. In *International conference on machine learning*, pages 8946–8970. PMLR, 2022.

[89] Ethan C Alley, Grigory Khimulya, Surojit Biswas, Mohammed AlQuraishi, and George M Church. Unified rational protein engineering with sequence-based deep representation learning. *Nature methods*, 16(12):1315–1322, 2019.

[90] Zuobai Zhang, Pascal Notin, Yining Huang, Aurélie Lozano, Vijil Chenthamarakshan, Debora Marks, Payel Das, and Jian Tang. Multi-scale representation learning for protein fitness prediction. *Advances in Neural Information Processing Systems*, 37:101456–101473, 2024.

[91] Roshan M Rao, Jason Liu, Robert Verkuil, Joshua Meier, John Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. Msa transformer. In *International conference on machine learning*, pages 8844–8856. PMLR, 2021.

[92] Céline Marquet, Julius Schlensok, Marina Abakarova, Burkhard Rost, and Elodie Laine. Expert-guided protein language models enable accurate and blazingly fast fitness prediction. *Bioinformatics*, 40(11):btae621, 11 2024. ISSN 1367-4811. doi: 10.1093/bioinformatics/btae621. URL https://doi.org/10.1093/bioinformatics/btae621.

[93] Sebastian Prillo, Wilson Wu, and Yun S Song. Ultrafast classical phylogenetic method beats large protein language models on variant effect prediction. *Advances in neural information processing systems*, 37:130265–130290, 2024.

Reproducibility Statement

We provide the source code of AUTOSCIENTISTS in https://github.com/mims-harvard/AutoScientists, which includes the launch scripts for all experiments. We use the official GitHub repositories for BioML-Bench (https://github.com/science-machine/biomlbench), ProteinGym (https://github.com/OATML-Markslab/ProteinGym), Autoresearch (https://github.com/karpathy/autoresearch), Biomni (https://github.com/snap-stanford/biomni), which we use with the base LLM Claude Sonnet 4.6, and Kermut (https://github.com/petergroth/kermut). We rerun MLAgentBench, AIDE, and STELLA on BioML-Bench with the base LLM GPT-5.4 and run the Codex variant of AUTOSCIENTISTS with the OpenAI Codex coding agent [63], and the repository includes the launch scripts for these runs. Further details are available in Table S1. We use only publicly available data in benchmarking from BioML-Bench, Kaggle (https://www.kaggle.com), TDC (https://tdcommons.ai), Polaris (https://polarishub.io), Open Problems (https://openproblems.bio), and ProteinGym (https://proteingym.org). For Autoresearch and AUTOSCIENTISTS we launch experiments with new Claude Code sessions with the prompt "Read the program file and start the experimental loop, do not stop until the specified wall clock time is up", and the AI agents work with no human intervention for the duration of the specified wall clock time.

Impact Statement

AUTOSCIENTISTS may accelerate machine learning and AI for Science by helping researchers explore modeling choices, use experimental compute more efficiently, and document both successful and failed hypotheses through shared logs, dead-end registries, model cards, and research reports. These benefits are most relevant for computational biomedical and scientific ML settings where experimental iteration is costly. The main risks are over-trusting automatically discovered models, overfitting to benchmark feedback, and amplifying erroneous hypotheses if validation is weak. This is important in biomedical applications, where outputs should not be treated as clinically or biologically actionable without expert review and independent validation. Our experiments are restricted to computational benchmarks on publicly available datasets, and deploying AUTOSCIENTISTS beyond these settings would require safeguards that we did not evaluate, including sandboxed code execution with audit logs, human approval for sensitive actions, hidden test sets, and independent replication.

A Implementation Details and Algorithmic Protocols

This section gives the implementation behind the method in Section 3. An AUTOSCIENTISTS run proceeds in three phases. In *Launch* (App. A.1), the system provisions a roster of experiment agents and analysts with no team assignments and posts a single discussion trigger to the forum $\mathcal{F}$. In *Self-organized team formation* (App. A.3), all agents enter a structured discussion in which they propose research directions, vote on proposals, and write the team roster R, and the same protocol reopens whenever an analyst detects stagnation. In the *Normal cycle*, once R exists, agents are repeatedly invoked through a fixed rotation. On each invocation, an analyst generates new proposals for the team queue via coverage audit, empirical priors ranking, and diversity rules (App. A.7), and an experiment agent claims a queued experiment, runs it, and applies a noise-aware promotion gate to decide whether to update the champion (App. A.6). All coordination flows through the shared state $\mathcal{S}$, which agents access through a list–decide–read protocol (App. A.4) and which contains per team queues Q_k governed by optimistic locking (App. A.5). Algorithms 1–4 give the per invocation pseudocode and are discussed in App. A.2.

A.1 Setup and Launch

An AUTOSCIENTISTS deployment is initialized by a deterministic bootstrap procedure that performs three steps. First, the shared state $\mathcal{S}$ is initialized with the task specification and an empty experiment log. Second, the agent roster is created, comprising experiment agents (each bound to one GPU if needed) and analysts, and each agent is issued a unique credential and a role-specific heartbeat description that determines its behavior at every invocation. The agents are launched with no team assignments, so the roster R that maps agents to teams is initially empty. Third, a single `[DISCUSSION-TRIGGER]` is posted to the forum $\mathcal{F}$, initiating the bootstrap discussion in which agents propose research directions and self-organize into teams (Section 3, App. A.3).

Table S1: Existing assets used in this work. We report the asset type, how it was used, citation, version or commit when available, license, and URL or terms of use.

Asset	Type	Use in this paper	Citation	Version / commit	License	URL
BioML-Bench	Benchmark / datasets / code	Evaluation on 24 biomedical ML tasks.	[2]	`673f3d8`	MIT	link
ProteinGym	Benchmark / datasets / code	Protein engineering tasks and 217-assay benchmark.	[17]	PG_v1.3	MIT	link
Autoresearch	Codebase / baseline	Single-agent baseline and GPT nanochat comparison.	[3]	`228791f`	MIT	link
Biomni	Codebase / baseline	Biomedical agent baseline on BioML-Bench.	[10]	0.0.8	Apache-2.0	link
MLAgentBench	Codebase / baseline	Baseline agent rerun on BioML-Bench.	[58]	`5d71205`	MIT	link
AIDE	Codebase / baseline	Baseline agent rerun on BioML-Bench.	[11]	`5d66a21`	MIT	link
STELLA	Codebase / baseline	Baseline agent rerun on BioML-Bench.	[5]	`1a8386e`	Apache-2.0	link
Kermut	Codebase / baseline model	Seed method for AUTOSCIENTISTS-Kermut.	[59]	`7e9e2e6`	MIT	link
Claude Code / Claude Sonnet 4.6	LLM / coding agent	Base coding agent backend.	[56, 57]	Claude Sonnet 4.6	API terms	link
OpenAI Codex	LLM / coding agent	Alternative coding agent backend (Appendix C.2).	[63]	GPT	API terms	link
Kaggle UW-Madison GI tract	Dataset	Biomedical imaging benchmark task.	[64]	Jul. 2022	Non-commercial academic research	link
Kaggle OSIC pulmonary fibrosis	Dataset	Biomedical imaging benchmark task.	[65]	Oct. 2020	Non-commercial academic research	link
Kaggle histopathologic cancer detection	Dataset	Biomedical imaging benchmark task.	[66]	Mar. 2019	Non-commercial academic research	link
Kaggle RSNA-MICCAI brain tumor	Dataset	Biomedical imaging benchmark task.	[67]	Oct. 2021	Non-commercial academic research	link
TDCommons	Dataset	Drug discovery benchmark tasks.	[68]	`c310c35`	MIT	link
Polaris Hub tasks	Dataset	Drug discovery benchmark tasks.	[69]	0.13.0	Apache-2.0	link
Open Problems	Dataset / benchmark	Single-cell omics benchmark tasks.	[70]	`5d53ffb`	MIT	link
ESM-2 models	Pretrained model	Embeddings and fine-tuning components.	[71]	650M, 3B	MIT	link
ProteinMPNN	Pretrained model / features	Conditional probability and structure kernel features	[72]	`8907e66`	MIT	link
ChemBERTa	Pretrained model	Molecular embeddings.	[73]	Zinc-Base-V1	MIT	link
Chemprop	Codebase / model	Models for BioML-Bench.	[74]	v2.2.3	MIT	link
RDKit	Software library	Molecular fingerprints and descriptors.	[75]	Q1 2026	BSD-3-Clause	link
XGBoost	Software library	Gradient-boosted tree models.	[76]	3.2.0	Apache-2.0	link
LightGBM	Software library	Gradient-boosted tree models.	[77]	4.6.0	MIT	link
CatBoost	Software library	Gradient-boosted tree models.	[78]	1.2.10	Apache-2.0	link
EfficientNet	Pretrained model	Embeddings and fine-tuning components.	[79]	B0, B3, B4	Apache-2.0	link
PyTorch / PyTorch Geometric	Software libraries	Neural network training and implementation.	[80]	2.11.0	BSD-3-Clause	link

After bootstrap, agents are repeatedly invoked in a fixed rotation by a simple loop. Each invocation is a single LLM session that wakes up, executes one heartbeat, and exits. Long-horizon coordination is achieved by repeating these invocations. The loop passes only the agent's identity, and the agent reads its persistent state (credentials, role assignment, and GPU binding for experiment agents) and the shared state $\mathcal{S}$ on every invocation to discover team membership, the current champion p^*, the queue state, and recent forum activity. Repeat discussion rounds are triggered by agents themselves when stagnation is detected (Appendix A.2).

Algorithm 1 Heartbeat dispatch (per agent invocation). The discussion branch on line 3 is detailed in Algorithm 2, and the role-specific normal cycle on line 9 is detailed in Algorithm 4 (analysts) or Algorithm 3 (experiment agents).

Require: agent identity i

1: Read roster R and recent forum $\mathcal{F}$ from $\mathcal{S}$
2: **if** $\mathcal{F}$ has an unresolved `[DISCUSSION-TRIGGER]` **or** R is empty **then**
3: **run** Algorithm 2; **return** ▷ discussion
4: **else if** i is not assigned to any team in R **then**
5: **return** ▷ no team exit
6: **else if** i is an experiment agent **and** an unposted result exists **then**
7: post pending `[RESULT]`; **return** ▷ resume
8: **end if**
9: Read $\mathcal{T}_k$, p^*, Q_k, knowledge files ▷ App. A.4
10: **run** Algorithm 4 if i is an analyst, **else** Algorithm 3 ▷ normal cycle
11: persist updated state and exit

Algorithm 2 Self-organized team formation (discussion branch). Detailed in App. A.3.

1: Read task specification, current champion p^* if any, and the active `[DISCUSSION-TRIGGER]` thread from $\mathcal{F}$
2: post candidate research directions to $\mathcal{F}$
3: post ranked hypotheses about the proposed directions, with reasoning
4: critique earlier posts and identify gaps in the proposed decomposition
5: post a self-termination vote on the trigger thread, either `[DISCUSS-MORE]` or `[DISCUSS-DONE]`
6: **if** a majority of agents have cast `[DISCUSS-DONE]` on the trigger thread **and** i is the alphabetically last analyst that participated **then**
7: consolidate proposals into a roster $R = \{(\mathcal{T}_k, \text{direction}_k, \text{members}_k)\}_{k=1}^{K}$
8: write R to $\mathcal{S}$
9: **end if**

Algorithm 3 Experiment agent cycle. Queue protocol in App. A.5 and noise-aware gate in App. A.6.

1: claim a queued experiment q from Q_k; **return** if none
2: apply diff to p^*, train candidate p', compute $\Delta = \ell(p') - \ell(p^*)$
3: **if** $\Delta > M\sigma$ **then**
4: promote p' to p^*
5: **else if** $0 < \Delta \leq M\sigma$ **then**
6: rerun on a second seed, and promote if both runs improve
7: **else**
8: discard
9: **end if**
10: record outcome to log $\mathcal{L}$, release claim on Q_k, post `[RESULT]` to $\mathcal{F}$

A.2 Heartbeat Protocol

Algorithm 1 specifies the per invocation dispatch, Algorithm 2 specifies what an agent does in the discussion branch, and Algorithms 4 and 3 specify the role-specific normal cycle. After reading the current roster and recent forum activity, the agent enters a discussion branch when an analyst has

Algorithm 4 Analyst cycle (when i is an analyst). Each step detailed in App. A.7.

if recent experiments produced no improvement, or proposals have concentrated on a narrow class of changes **then**
    post `[DISCUSSION-TRIGGER]` to $\mathcal{F}$ ▷ App. A.3
end if
audit untested parameters of p^*
compute research direction priors $\mu_{a,d}$ from experiment log $\mathcal{L}$
propose 2 experiments under ambition quota and diversity rules, and if p^* changed since the analyst's last cycle, ≥ 1 proposal must target the property responsible
append proposals to Q_k ▷ App. A.5

posted a `[DISCUSSION-TRIGGER]` in response to stagnation (Algorithm 4) or when the roster has not yet been formed during cold start bootstrap; a no team exit when the agent is not assigned to any team; a resume branch when an experiment agent has an unposted result from a prior session; or the role-specific normal cycle. In the normal cycle, the two roles divide labor, where an experiment agent claims a queued experiment, applies the code change, trains, gates the result against the noise floor, and records the outcome (Algorithm 3), while an analyst, rather than running experiments, maintains the team's knowledge by auditing untested parameters, ranking research directions by empirical effect size, and queueing new proposals for experiment agents to execute (Algorithm 4). In the discussion branch (Algorithm 2), both roles contribute research directions, hypotheses, and a self-termination vote, and the alphabetically last analyst that participated in the discussion is additionally responsible for consolidating the proposals into the new roster R (App. A.3). After the branch returns, the agent persists its updated state and exits.

A.3 Self-Organized Team Formation

The roster $R = \{(\mathcal{T}_k, \text{direction}_k, \text{members}_k)\}_{k=1}^{K}$ is not specified by an external coordinator, and agents write it themselves through a structured discussion that opens whenever the discussion branch of Algorithm 1 is entered. The same protocol applies in *cold start bootstrap* at the beginning of a run, when R is empty and every agent is routed to the discussion branch, and in *mid-run reformation*, when an analyst's stagnation check (Algorithm 4) posts a fresh `[DISCUSSION-TRIGGER]` to $\mathcal{F}$ and subsequent agents observe it.

Discussion contributions. Each agent entering the discussion branch reads the task specification, the current champion p^* if any, and the active `[DISCUSSION-TRIGGER]` thread. It then posts (i) candidate research directions, (ii) hypotheses ranking those research directions by expected effect with the reasoning that justifies the ranking, and (iii) a self-termination vote on the trigger thread, either `[DISCUSS-MORE]` (more discussion is needed before a roster can be written) or `[DISCUSS-DONE]` (the discussion has converged). Later agents critique earlier posts, identify gaps, propose alternative groupings of research directions, and add their own vote.

Termination and roster writing. The discussion terminates when a majority of agents have cast `[DISCUSS-DONE]` on the trigger thread. The alphabetically last analyst that participated in the discussion is responsible for consolidating the proposals into the new roster R and writing it to $\mathcal{S}$. The alphabetical convention is a tiebreaker that ensures exactly one agent assumes the consolidator role without further coordination, and the system has no privileged consolidator agent.

Reformation outcomes. A new roster may create, merge, split, retire, or rebalance teams relative to the previous roster, depending on the discussion's conclusions. Teams whose research directions have produced no recent improvements can be retired, newly discovered productive research directions can spawn new teams, and agents can be reassigned across teams. The persistence of R in $\mathcal{S}$ ensures that subsequent invocations begin from the written decomposition.

A.4 File Discovery

In each heartbeat cycle, agents access the shared state $\mathcal{S}$ through a list–decide–read protocol, where the agent first retrieves only the lightweight metadata for items in $\mathcal{S}$ (path, version, timestamp, author), then selects the items relevant to its current task, and finally fetches the contents of those items. This

indirection allows new artifact types to appear in $\mathcal{S}$ over the course of a run (for instance, a newly created knowledge file or dead-end registry) without changes to the access protocol.

A.5 Queue and Claim Protocol

The team queue Q_k is a structured record of pending experiments (each with an identifier, priority, diff description, proposing agent, and link to its proposal post) and the experiments currently claimed by agents. Concurrent reads and writes are serialized by optimistic locking on a version token, so a stale write is rejected and the agent retries against the latest version. Each queue update is therefore atomic and no experiment is claimed twice. Each experiment agent releases its own claim immediately after recording the experiment's outcome.

A.6 Noise-Aware Champion Validation

The evaluation metric ℓ is stochastic, since running the same program with two different random seeds yields different values. A naive rule that promotes whenever $\ell(p') > \ell(p^*)$ would therefore promote candidates whose "improvement" is just noise. All agents build on the shared champion p^*, so an erroneous promotion compounds because every downstream comparison is then made against a corrupted baseline. We term this *champion pollution* and prevent it by gating every proposed promotion against an empirically measured noise floor.

For a candidate p' produced by an experiment agent (Algorithm 3), let $\Delta = \ell(p') - \ell(p^*)$ denote its signed improvement over the current champion. Since ℓ is oriented so that higher is better (Section 3), $\Delta > 0$ if p' is better. Let σ denote the per run standard deviation of ℓ at fixed code (the *noise floor*, with calibration described below), and let $M = 2$, so that $[0, M\sigma]$ is the band of improvements small enough to be confounded with noise. The gate decides

$$\text{promote}(p') = \begin{cases} \texttt{true} & \text{if } \Delta > M\sigma \\ \text{confirm}(p', \text{seed}_2) & \text{if } 0 < \Delta \le M\sigma \\ \texttt{false} & \text{if } \Delta \le 0, \end{cases} \tag{1}$$

where $\text{confirm}(p', \text{seed}_2)$ reruns p' with a new random seed and returns `true` only if both runs improve over p^*. Improvements above the noise band are accepted directly, improvements inside the noise band are accepted only if a second seed rerun confirms, and non-improvements are rejected. Candidates that fall in the noise band but fail confirmation are recorded as *near-misses*.

The noise floor σ is calibrated lazily, without spending experiments specifically for calibration. Until σ has been estimated, the gate uses a conservative default noise band so that no candidate can be promoted on weak evidence. Each time the gate's middle branch fires, the resulting pair of duplicate seed measurements $(\ell_{1,i}, \ell_{2,i})$ for the same candidate code is recorded. Once at least three such pairs have accumulated ($n \ge 3$), σ is set to the within pair pooled standard deviation

$$\sigma = \sqrt{\frac{1}{2n} \sum_{i=1}^{n} (\ell_{1,i} - \ell_{2,i})^2}, \tag{2}$$

which estimates the per seed noise of ℓ at fixed code without conflating it with code-to-code variation. To prevent later pairs from retroactively reclassifying earlier promotion decisions, σ is locked once five pairs have been recorded.

A.7 Analyst Proposal Protocol

On each invocation, an analyst generates two new experiment proposals for the team queue through the following cycle.

The analyst first builds a working picture of what has been tried and what has not. It scans the champion code p^* for all numeric parameters (top-level constants, class fields, and inline numeric literals) and matches each against the experiment log $\mathcal{L}$ to identify parameters that have never been varied (*baseline coverage audit*). It then computes the empirical mean effect size per (axis, direction) pair from prior experiments,

$$\mu_{a,d} = \frac{1}{|E_{a,d}|} \sum_{e \in E_{a,d}} |\Delta_e|, \tag{3}$$

designates research directions with $|E_{a,d}| < 3$ as *cold* and gives them an exploration bonus, deprioritizes directions whose effect size falls below the noise floor ($\mu_{a,d} < \sigma$), and sorts the team queue Q_k by the resulting ranking (*empirical research direction priors*).

The analyst then drafts two new proposals subject to two filters. The *ambition quota* requires at least one of the two to satisfy a bold move criterion, such as a parameter count change of at least 10%, a fix to a confirmed bug in the champion code, an experiment on a research direction flagged as untested in two or more prior discussion threads, or a hypothesis tension probe whose outcome will confirm or falsify the team's current hypothesis. If neither proposal meets this criterion, the analyst must post a public `[EXEMPT]` comment justifying why no bold candidate exists, rather than accept incrementalism. The *diversity constraints* additionally require that (i) the two proposals target different research directions, (ii) no three or more consecutive recent proposals on the same axis push in the same direction, and (iii) no proposal falls inside a previously rejected range in the dead-end registry $\mathcal{D}_k$ unless the proposer explicitly states what differs from the prior failure.

When the champion has been updated since the analyst's previous cycle (a *KEEP*, which is a successful promotion of a candidate to a new p^*), a final *post KEEP* step requires at least one of the two proposals to follow up on the property responsible for the recent improvement via a different mechanism. To enforce this, the analyst must answer which property of the successful change made it work and what other untried changes share that property. The completed proposals are appended to Q_k.

B Extended Ablation Results

The main paper ablation table (Table 3) reports the three single component removals (`No analyst`, `No cross-agent`, `No self-org.`) and the `Independent agents` variant against the full AUTOSCIENTISTS system. We report three additional comparisons on the same task suite here, (i) AUTOSCIENTISTS versus autoresearch [3] as a single-agent reference (Table S2 for GPT training optimization and Table S10 for BioML-Bench), (ii) two variants of the no self-organization ablation that change only the initial teams (Table S3), and (iii) a team size sweep with working agent counts $n \in \{2, 4, 14\}$ against the default crew ($n = 9$) (Table S4).

Table S2: **AUTOSCIENTISTS versus Autoresearch** [3] on GPT Training Optimization across two settings.

Setting	Metric	Start `val_bpb`	# Experiments	Autoresearch	AUTOSCIENTISTS
From Autoresearch baseline	Best `val_bpb` (↓)	0.998	50	0.9790	0.9777
From AUTOSCIENTISTS champion	Best `val_bpb` (↓)	0.9777	100	0.9777 (0 KEEPs)	**0.9730** (7 KEEPs)

B.1 Comparison Against Autoresearch at Matched and Extended Compute

Table S2 reports two settings. (i) *From the Autoresearch baseline*, both systems start from the original GPT training setup released with Autoresearch [3] (the unmodified `nanochat` code at `val_bpb` $=$ 0.998) and run 50 experiments, and AUTOSCIENTISTS reaches 0.9777 whereas Autoresearch reaches 0.9790. (ii) *From the same* AUTOSCIENTISTS *champion* at `val_bpb` $=$ 0.9777 (the result of AUTOSCIENTISTS's setting (i) run) for 100 additional experiments, AUTOSCIENTISTS accepts seven KEEPs and reaches 0.9730, while Autoresearch accepts zero KEEPs and produces no improvement over the starting champion. AUTOSCIENTISTS's advantage over the single-agent loop grows with the number of experiments, because the single-agent loop saturates while AUTOSCIENTISTS continues to find productive directions.

B.2 Separating Reorganization from Initial Teams

The no self-organization ablation in Table 3 differs from AUTOSCIENTISTS in that its teams are predefined at the start of the run and cannot be reorganized during the run. To separate these factors, we run two further variants on GPT nanochat training optimization. Both keep the same agent roster, parallel execution, shared state (champion, experiment log, and dead-end registry), and cross-agent feedback as AUTOSCIENTISTS, keep mid-run reorganization disabled, and use about 50 experiments, beyond the point at which AUTOSCIENTISTS reaches its final `val_bpb` (by experiment ≈45, and ≈0.978 by experiment 34). In the first variant, agents discuss and form their own initial teams, and

these teams are then kept fixed. In the second, the run is seeded with the exact teams and hypotheses that AUTOSCIENTISTS self-organized, and only mid-run reorganization is disabled. We verified that these runs contain no discussion posts and that the roster stays fixed throughout.

As shown in Table S3, self-organized initial teams improve over predefined initial teams (0.9806 vs. 0.9833), so self-organization at the start of the run contributes, but it does not close the gap to AUTOSCIENTISTS. With the initial teams copied exactly from AUTOSCIENTISTS, the fixed team variant still reaches only 0.9794, compared with 0.9777 for the full system. Because the initial teams, parallelism, shared state, and feedback are matched, this remaining gap is consistent with mid-run reorganization contributing beyond better initial teams.

To check that these differences exceed seed variance, we reran five configurations with independent random seeds and estimated a per seed standard deviation of $\sigma \approx 0.0007$ `val_bpb`. The gaps between AUTOSCIENTISTS and each ablation in Tables 3 and S3 exceed the corresponding 2σ band (≈ 0.0015), although each condition is a single run.

Table S3: **Separating mid-run reorganization from the initial teams on GPT nanochat training optimization.** All variants keep parallel agents, the shared state, and cross-agent feedback, and the three no self-organization variants disable mid-run reorganization and differ only in how the initial teams are formed. The per seed standard deviation is $\sigma \approx 0.0007$ `val_bpb`.

Condition	**Best** `val_bpb` (↓)
No self-organization, predefined initial teams (Table 3)	0.9833
No self-organization, agent discussed initial teams, then fixed	0.9806
No self-organization, exact AUTOSCIENTISTS initial teams, then fixed	0.9794
AUTOSCIENTISTS	**0.9777**

Table S4: **AUTOSCIENTISTS agent team size sweep.** Three tasks, two of them from Table 3, comparing the default AUTOSCIENTISTS crew ($n = 9$) against three crew size variants ($n = 2$, $n = 4$, $n = 14$). n is the number of experiment and analyst agents. Bold indicates the best result per row. SPIKE-SARS2 here is the cold start BioML-Bench task, in which agents receive no prior model, and differs from the Kermut seeded setting of Sec. 4.4. Its n=9 entry is the first of three independent runs, whose mean is 0.663 ± 0.025 (Appendix C.2).

		AUTOSCIENTISTS agent team size			
Task	**Metric**	**n=2**	**n=4**	**n=9 (default)**	**n=14**
TDC-hERG	AUROC (↑)	0.780	0.803	**0.867**	0.843
	Leaderboard % (↑)	14.3	14.3	**85.7**	42.9
BioML-Bench SPIKE-SARS2 (cold start)	Spearman's ρ (↑)	**0.874**	0.835	0.670	0.506
	Leaderboard % (↑)	**100.0**	**100.0**	81.8	72.7
GPT Training Optimization	Best `val_bpb` (↓)	**0.9777**	0.9778	**0.9777**	0.9821

B.3 Team Size Sensitivity, Parallelism Improvement, and Oversubscription

We compare four working crew sizes (n=2, 4, 9, 14) on three tasks (TDC-hERG, BioML-Bench SPIKE-SARS2 (cold start), GPT nanochat training optimization). Crew composition (experiment agents + analysts) is n=2 (1+1), n=4 (2+2), n=9 (6+3, the default AUTOSCIENTISTS configuration), n=14 (9+5).

Crew size sensitivity is task dependent. The three tasks in Table S4 show different sensitivities to crew size, and the crew size that achieves the top score differs across tasks. On TDC-hERG, AUROC spans 0.780–0.867 and leaderboard percentile spans 14.3–85.7 across the four crews. On BioML-Bench SPIKE-SARS2, Spearman ρ spans 0.506–0.874 and leaderboard percentile spans 72.7–100.0. On GPT training optimization, `val_bpb` spans 0.9777–0.9821. At n=14, every task degrades relative to its best score in the table. These results suggest that the optimal crew size varies across tasks rather than being a fixed property of the protocol. Performance is not monotonic in the number of agents, since on SPIKE-SARS2 the two smallest crews outperform the default n=9 crew, and the largest crew is never the best. Nine agents is therefore not optimal for every task, although every crew of up to nine agents outperforms single-agent Autoresearch on SPIKE-SARS2 (0.593, Table S10). Dynamically scaling team size to task difficulty is a direction for future work.

Choosing the number of agents in practice. The value of additional agents depends on how decomposable the task is, the coordination structure, the capability of the base model, and how many distinct research directions are useful. The current implementation adapts the organization of a fixed pool of agents but does not change the number of active agents, so we recommend selecting the agent count for each task and budget. First, identify how many experiments can run in parallel and use enough experiment agents to occupy, but not exceed, the available experimental compute. Then choose the number of analysts with a small pilot sweep, taking the smallest configuration beyond which performance or proposal diversity no longer improves. Duplicate proposals, rediscovered dead ends, idle proposal queues, and rising communication cost without new champion improvements indicate that the team is too large. Unexplored directions, or an experiment queue dominated by a narrow family of ideas, suggest adding analysts.

Parallel execution of more agents is faster at similar quality. AUTOSCIENTISTS agents are designed to run in parallel, and within each heartbeat all n working agents fire one LLM call simultaneously. Due to resource limits in our setup, we executed the agents iteratively rather than fully in parallel, so we report two complementary views. The per experiment axis (Fig. S1) compares final quality under matched experimental compute, and the three smaller crews converge to similar final `val_bpb` ($\sim$ 0.9777, within the noise floor) over 51, 38, and 71 experiments for n=2, 4, 9 respectively. The per heartbeat axis (Fig. S2) projects the wall-clock cost of the same runs under fully parallel execution, with $\approx$ 26, $\approx$ 10, and $\approx$ 8 heartbeats for n=2, 4, 9, a $\sim 3.25\times$ speedup of n=9 over n=2. Per experiment quality is therefore similar across crews in the productive range, while parallel execution lets larger crews finish faster. These heartbeat counts are a projection of agent cycles under ideal parallelism, not a measurement of GPU wall-clock time.

Measured wall-clock under parallel GPU execution. To measure wall-clock time under true parallel execution, we fix the agent team, base model, and GPT nanochat training setting, and vary only the number of GPUs that run experiments in parallel ($P \in \{1, 2, 4\}$). Table S5 reports the average wall-clock time to complete one experiment after five full agent cycles. Wall-clock time per experiment decreases monotonically with more GPUs, reaching a $2.18\times$ speedup at P=4. The speedup is sublinear because each experiment involves only a few minutes of GPU training relative to the LLM reasoning and I/O of the agents, and this fixed per experiment overhead does not shrink with the number of GPUs. This overhead is a property of the implementation rather than of the coordination method, and it could be reduced with asynchronous dispatch, in which each freed GPU immediately claims the next queued experiment.

Table S5: Measured wall-clock time per experiment on GPT nanochat when experiments run on P GPUs in parallel, averaged after five full agent cycles.

GPUs (P)	Wall-clock per experiment	Speedup vs. P=1
1	9.7 min	1.00×
2	6.4 min	1.52×
4	4.4 min	2.18×

C AUTOSCIENTISTS Performance Across Independent Runs

C.1 GPT nanochat training optimization

We executed three independent cold start runs of AUTOSCIENTISTS on the GPT nanochat task to assess whether AUTOSCIENTISTS's behavior is stable across the stochastic factors that arise during cold start coordination (workshop post acquisition order, concurrent claim races, per experiment random seeds). The three runs use identical hardware, templates, and task specification, and none uses a fixed seed or deterministic replay mechanism, so each constitutes an independent realization.

Figure S3 reports the running best-so-far `val_bpb` for all three runs. All three runs converge to the same best-so-far region, with final values of 0.9777 (Run 1), 0.9795 (Run 2), and 0.9780 (Run 3) after 75, 64, and 62 experiments respectively. The mean final `val_bpb` across runs is 0.9784 with sample standard deviation 0.0010 (sample variance 9.4×10^{-7}, range 0.0018). Acceptance rates are comparable across runs, at $8/75 = 10.7\%$ for Run 1, $7/64 = 10.9\%$ for Run 2, and $6/62 = 9.7\%$ for Run 3. The runs select different intermediate research directions (Newton–Schulz iteration count and

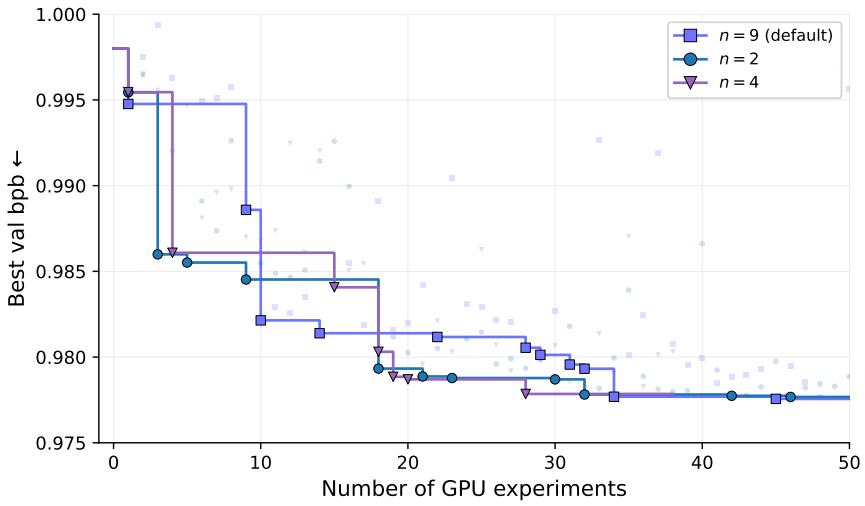


Figure S1: Team size sweep on GPT nanochat as a function of *experiment number*. Solid step curves are the running best-so-far validation `val_bpb`, opaque markers indicate accepted KEEPs, and transparent markers show per experiment attempts that did not improve the running min. Final values are n=2 0.9777 (11 KEEPs / 51 exps), n=4 0.9778 (7 KEEPs / 38 exps), default AUTOSCIENTISTS (n=9) 0.9777 (11 KEEPs / 71 exps), n=14 0.9821 (15 KEEPs / 50 exps).

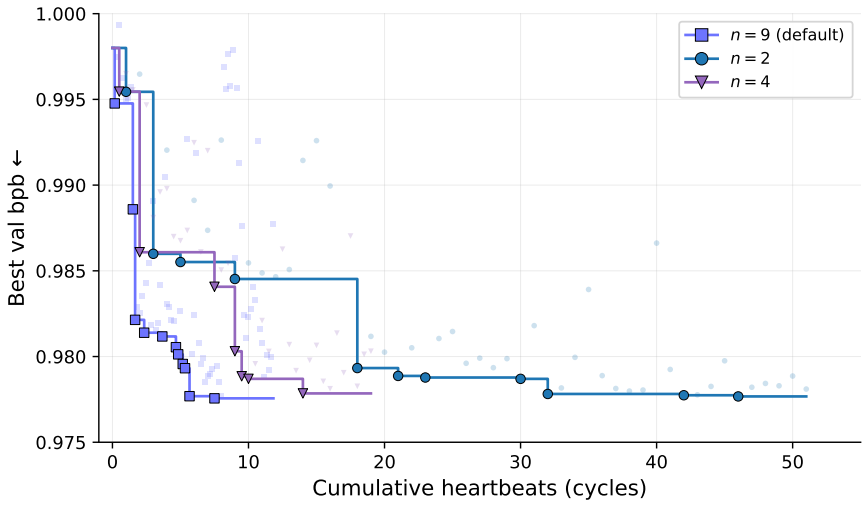


Figure S2: Same trajectories as Fig. S1, plotted against *cumulative heartbeats* under fully parallel execution (one heartbeat = one synchronized cycle in which all n working agents fire one LLM call simultaneously). n=2 finishes in ≈26 heartbeats, n=4 in ≈10, default n=9 in ≈8, and n=14 in ≈4. Larger crews finish faster under parallel execution.

short window patterns in Run 1, final LR fraction and per group learning rate allocation in Run 2, and total batch size halving and rotary base scaling in Run 3) yet reach the 0.978–0.980 region, indicating that different proposal orderings reach the same productive region rather than depending on a single deterministic path. Descent timing differs across runs because each run is gated by a small number of high leverage probes whose discovery order is stochastic. Run 3 trails Runs 1–2 through the middle range, recovers at experiment 37 once a coupling preserving warmdown ratio probe lands, and then makes a large late jump (a width up KEEP at experiment 43 worth −0.0048 `val_bpb`), reaching its final region within five experiments. The 0.0018 gap between Runs 2 and 1 reflects a single `depth_7` schedule probe that Run 1 reached at experiment 54 and Run 2 was halted before reaching, while Run 3 closes most of the same gap through an alternate path.

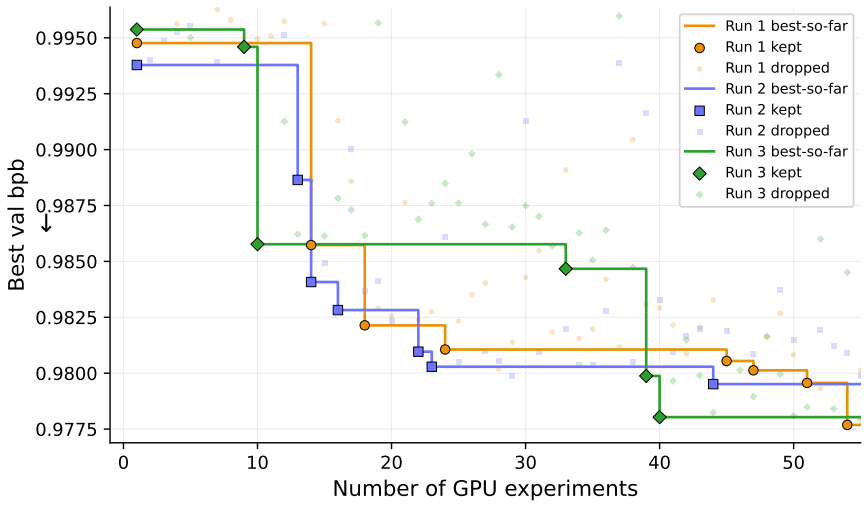


Figure S3: Per experiment trajectories of three independent cold start runs of AUTOSCIENTISTS on the GPT nanochat task. Solid step curves show the running best-so-far validation `val_bpb`, opaque markers indicate accepted KEEPs, and transparent markers show per experiment attempts that did not improve the running minimum. All three runs converge to the same best-so-far region under different proposal orderings, with descent timing that depends on when high leverage probes are discovered.

C.2 Independent run stability across tasks and LLM backends

Table S6 reports repeated runs of AUTOSCIENTISTS on TDC-hERG with the Claude Code and OpenAI Codex backends, on GPT nanochat, and on SPIKE-SARS2 in the cold start BioML-Bench and Kermut seeded settings. The other BioML-Bench tasks were each run once.

SPIKE-SARS2 Kermut seeded. Three independent runs from the same Kermut seed reach 0.8377 ± 0.0037, compared with 0.8088 ± 0.0058 for three single-agent Autoresearch runs given the same seed, experimental compute, and number of experiments. All three AUTOSCIENTISTS runs converge on the same two modifications, neither of which is present in Kermut. First, they broaden the zero-shot evidence supplied to the Gaussian process, since Kermut uses a single zero-shot predictor whereas the three discovered models use fifteen, seven, and seven predictors, as kernel features, linear mean inputs, or both. Second, they replace Kermut's plain z-scoring of targets with a nonlinear distributional transformation (quantile-based normalization, rank-based normalization, and Yeo–Johnson, respectively). Autoresearch also discovers the broadened zero-shot features but none of the target transformations, reaching 0.8051, 0.8059, and 0.8155 in its three runs.

LLM backends. To test whether AUTOSCIENTISTS depends on the Claude Code backend, we replace it with the OpenAI Codex coding agent [63], keeping the rest of the system unchanged. On GPT nanochat training optimization, starting from the Autoresearch baseline (`val_bpb` $= 0.998$), a single Codex run of 100 experiments reaches a best `val_bpb` of 0.9765 by experiment 19 and makes no further improvement, compared with 0.9777 for Claude backed AUTOSCIENTISTS and 0.9790 for Autoresearch within the 50-experiment budget of Table S2. This run is not matched in the number of experiments to the other two, so it indicates only that AUTOSCIENTISTS performs comparably with either backend. On TDC-hERG, three independent Codex runs reach a mean AUROC of 0.864 ± 0.012, compared with 0.853 ± 0.020 with Claude Sonnet 4.6 (Table S6), and this mean is above the single runs of Biomni (0.855) and Autoresearch (0.800) (Table S10).

Across tasks, repeated runs identify similar high-level strategies but differ in the specific models and implementations they select, and hence in final score.

Table S6: Repeated independent runs of AUTOSCIENTISTS, reported as mean ± sample standard deviation over three runs.

Task / setting	Metric	Run 1	Run 2	Run 3	AUTOSCIENTISTS
TDC-hERG (Claude Code, Sonnet 4.6)	AUROC (↑)	0.867	0.830	0.862	0.853 ± 0.020
TDC-hERG (OpenAI Codex)	AUROC (↑)	0.859	0.855	0.877	0.864 ± 0.012
GPT nanochat (cold start)	`val_bpb` (↓)	0.9777	0.9795	0.9780	0.9784 ± 0.0010
BioML-Bench SPIKE-SARS2 (cold start)	Spearman's ρ (↑)	0.6701	0.6351	0.6845	0.6632 ± 0.0254
SPIKE-SARS2 (Kermut seeded)	Spearman's ρ (↑)	0.8407	0.8389	0.8336	0.8377 ± 0.0037

D Per Experiment Trajectories for Ablation Runs

This appendix plots per experiment trajectories for the ablation runs reported in Table 3. All ablations start from the pristine nanochat baseline (`val_bpb` $= 0.998$) and report the running best-so-far validation `val_bpb`. Transparent markers show every attempted experiment, and opaque markers indicate accepted improvements (KEEPs). Autoresearch [3] and the full AUTOSCIENTISTS system on the same anchor are included in each plot for reference. The ablation runs use a 50-experiment budget, and the no self-organization run is additionally extended to 71 experiments. The reference trajectories are longer (71 experiments for the full system and 83 for Autoresearch), so their final values are not a controlled terminal comparison, and their difference (0.9777 vs. 0.9773) is smaller than the per seed noise of `val_bpb` ($\sigma \approx 0.0007$, Appendix B.2). The controlled comparison between AUTOSCIENTISTS and Autoresearch at 50 experiments is in Table S2. For each run we distinguish the total run length from the experiment at which the final best value is reached.

D.1 No Self-Organization (`abl-no-self-org`)

Removing self-organization (teams are fixed at the start of the run, with no mid-run reformation) reduces the system to 5 KEEPs and a final `val_bpb` of 0.9833 over 71 experiments, with no further improvement after its last KEEP (Fig. S4). The full AUTOSCIENTISTS system on the same anchor reaches 11 KEEPs and 0.9777 in 71 experiments, reaching 0.9777 by experiment ≈45 and ≈0.978 by experiment 34. The ablated system tracks the full system for the first ∼10 experiments (when initial high effect size research directions are still being swept) but stalls thereafter, consistent with `ISSUES.md`, which shows that hypothesis falsification triggers fired (analyst1 on arch v3, analyst2 on optim v3) but could not take effect without team reformation. We extended this run from 47 to 71 experiments, the same total length as the full system run, and it made no further improvement, so the gap is not caused by a difference in run length.

D.2 No Cross-Agent Communication (`abl-no-cross-agent`)

Removing cross-agent communication leaves coordination only through the champion record, dead-end registry, team queue, and result log. Only `[PROPOSAL]` and `[RESULT]` posts are allowed, and all other forum activity, including comments, notifications, gap analyses, rankings, synthesis posts, near-miss reports, suggestions, and structural change proposals, is disabled, as is the stagnation triggered discussion mechanism. The resulting system reaches 9 KEEPs and a final `val_bpb` of 0.9814 over 50 unique experiments (Fig. S5), against 11 KEEPs and 0.9777 in 71 experiments for the full system on the same anchor. Unlike the no self-organization ablation, the no cross-agent run continues to find improvements throughout the budget, since the shared artifacts alone carry enough signal for decisions within individual research directions to converge to a similar neighborhood. The cost is throughput, since the run needed roughly $1.85\times$ more experiments than a comparable full system run (measured against an internal full system run from the pristine baseline that reached 0.980289 in 27 experiments) and never bridged the last ∼0.001 gap, because teams duplicated brackets that cross-team comments and team restructuring proposals would otherwise have deduplicated. The single largest individual win in this run was `sched_batch_half` (TOTAL_BATCH_SIZE $2^{19} \to 2^{18}$, $\Delta = -0.0067$), which agents discovered independently after several teams converged on a step limited diagnosis from the artifact log alone.

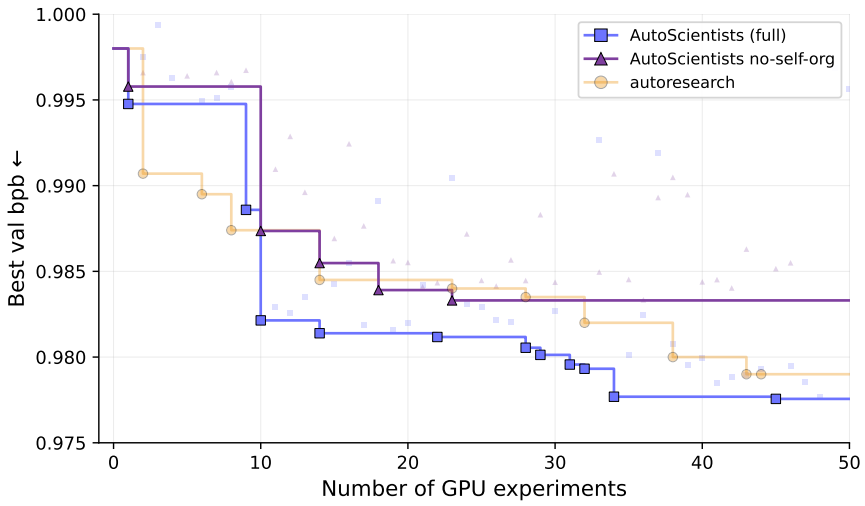


Figure S4: Per experiment trajectory for the `abl-no-self-org` ablation compared with the full AUTOSCIENTISTS system and single-agent autoresearch, all anchored at the pristine nanochat baseline (`val_bpb` = 0.998). Solid step curves are the running best-so-far validation `val_bpb`, opaque markers indicate accepted KEEPs, and transparent markers show per experiment attempts that did not improve the running min. Final values are autoresearch 0.9773 (15 KEEPs / 83 exps), full AUTOSCIENTISTS 0.9777 (11 KEEPs / 71 exps), `abl-no-self-org` 0.9833 (5 KEEPs / 71 exps). The plot shows the first 50 experiments of each trajectory. The no self-organization run was extended to 71 experiments, the same length as the full system run, and made no further improvement after the last KEEP shown here.

D.3 No Analyst Role (`abl-no-analyst`)

Removing the analyst role (no analyst agents at all, and experiment agents handle proposal generation, research direction prioritization, and post KEEP induction in addition to executing experiments) leaves the system with five experiment agents operating against the shared workshop and artifact surface alone. The resulting system reaches 7 KEEPs and a final `val_bpb` of 0.9817 over 50 unique experiments (Fig. S6), against 11 KEEPs and 0.9777 in 71 experiments for the full system on the same anchor. In this run, experiment agents self-organized cross-team probes when their team's native research directions were exhausted. After the schedule team closed its TOTAL_BATCH / WARMUP / WARMDOWN / FINAL_LR_FRAC bracket, gpu3 rebracketed the architecture team's MLP_RATIO under the new compute regime (yielding KEEPs at MLP=5 and MLP=6) and gpu6 probed the optimizer team's UNEMBEDDING_LR under the new champion (yielding a third cross-team KEEP). Three of the seven accepted improvements thus came from experiment agents probing research directions another team had closed or skipped, and the shared workshop and the post KEEP cross-team suggestion threads carried the coordination signal that an analyst would otherwise have written. Experiment agents can therefore take over part of the analyst's coordination role, but not all of it. Although each ablation condition is a single run, the gap to the full system (0.9817 vs. 0.9777) is larger than twice the per seed noise of `val_bpb` ($2\sigma \approx 0.0015$, Appendix B.2) and than the spread of three independent full system runs (0.9777–0.9795, Appendix C).

D.4 Independent Agents (`abl-independent`)

Removing all cross-agent coordination (no comments, no notifications, no suggestions, near-miss reports, discussion posts, gap analyses, or rankings, and no shared artifacts) and reducing the roster to six experiment agents leaves each agent running an independent autoresearch loop with its own private champion seeded from the pristine upstream and no view of any other agent's state. The resulting system reaches a best-of-population `val_bpb` of 0.9833 over 50 unique experiments (Fig. S7), against 11 KEEPs and 0.9777 in 71 experiments for the full system on the same anchor. Whereas `abl-no-cross-agent` forbids communication between agents but still lets them inherit each other's wins through a shared champion program and dead-ends file, this ablation additionally hides those files. The cost of this further isolation is visible in two ways. First, five of the six agents independently

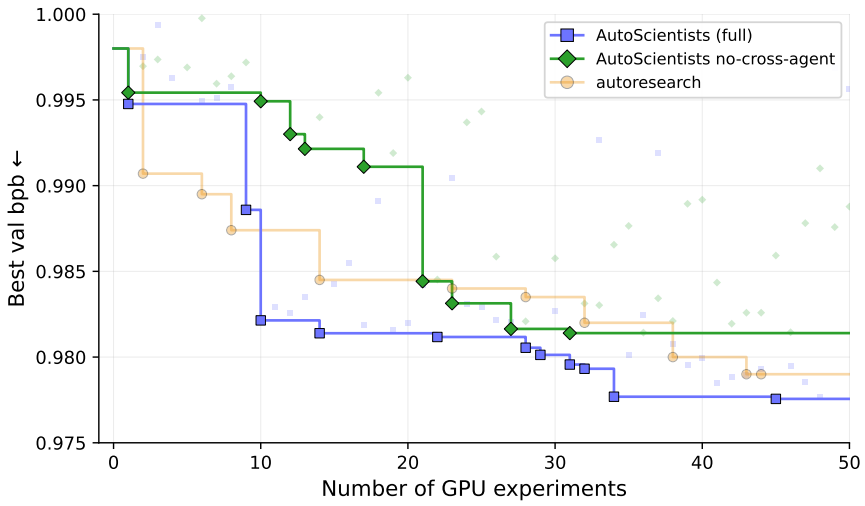


Figure S5: Per experiment trajectory for the `abl-no-cross-agent` ablation compared with the full AUTOSCIENTISTS system and single-agent autoresearch, all anchored at the pristine nanochat baseline (`val_bpb` $= 0.998$). Solid step curves are the running best-so-far validation `val_bpb`, opaque markers indicate accepted KEEPs, and transparent markers show per experiment attempts that did not improve the running min. Final values are autoresearch 0.9773 (15 KEEPs / 83 exps), full AUTOSCIENTISTS 0.9777 (11 KEEPs / 71 exps), `abl-no-cross-agent` 0.9814 (9 KEEPs / 50 exps). The ablation uses a 50-experiment budget, while the full system and Autoresearch curves are longer reference trajectories and are not a controlled terminal comparison.

rediscovered the same dominant first improvement (the `TOTAL_BATCH_SIZE` $2^{19} \rightarrow 2^{18}$ reduction also surfaced in `abl-no-cross-agent`) within their first two experiments, spending roughly a third of the budget on duplicates that the shared dead-ends ledger would have suppressed. Second, the agents that escaped the pristine plateau followed disjoint search paths and produced incompatible champions. Without shared artifacts, the population cannot combine these disjoint wins, and the best-of-population trajectory plateaus roughly 0.006 above the full system anchor.

E AUTOSCIENTISTS Output on GPT nanochat

On the GPT nanochat training optimization task, AUTOSCIENTISTS descended from the Autoresearch baseline at `val_bpb` $= 0.998$ to a final champion at `val_bpb` $= 0.97769$ over 75 experiments. Alongside the champion training script, the run produces two structured documents that follow the templates released with the system, a *model card* (Sec. E.1) covering model details, training procedure, evaluation, and compute, and a *research insights* document (Sec. E.2) recording the reasoning trajectory, including which research directions were tried, which were accepted, which were rejected, and what the analyst notes recorded as the mechanism in each case. The per experiment trajectory is the Run 1 trace in Figure S3.

E.1 Model Card

This card is populated from the champion record. Fields that are not informative at this scale (carbon emission accounting, demo links, downstream use guidance) are omitted. An analogous card for a BioML-Bench task is shown in Figure 2 of the main paper.

Model details.

- **Developed by:** AUTOSCIENTISTS multi-agent system (6 experiment and 3 analyst agents), with Claude Sonnet as the per agent backend.
- **Model type:** decoder-only transformer language model, 87 M parameters.
- **Language:** English (FineWeb shard).
- **Base model:** pristine nanochat baseline released with Autoresearch [3] (`val_bpb` $= 0.998$).

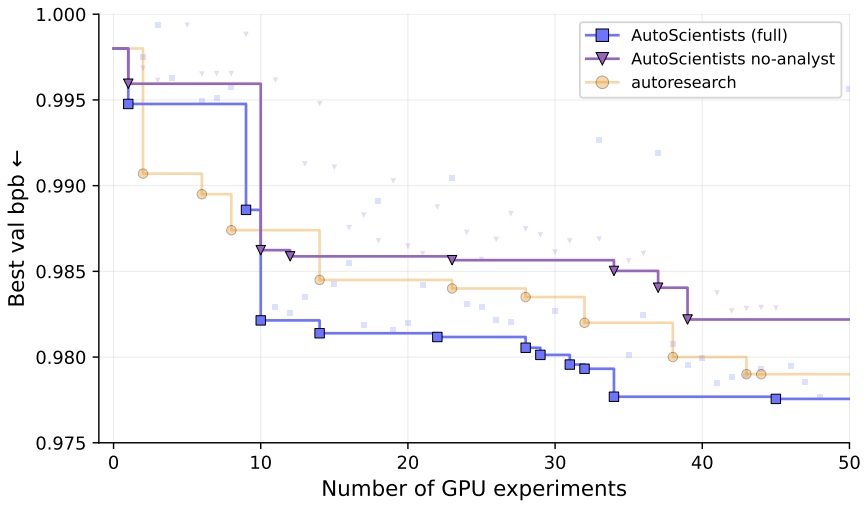


Figure S6: Per experiment trajectory for the `abl-no-analyst` ablation (purple) compared with the full AUTOSCIENTISTS system (red) and single-agent autoresearch (blue), all anchored at the pristine nanochat baseline (`val_bpb` $= 0.998$). Solid step curves are the running best-so-far validation `val_bpb`, opaque markers indicate accepted KEEPs, and transparent markers show per experiment attempts that did not improve the running min. Final values are autoresearch 0.9773 (15 KEEPs / 83 exps), full AUTOSCIENTISTS 0.9777 (11 KEEPs / 71 exps), `abl-no-analyst` 0.9817 (7 KEEPs / 50 exps). The ablation uses a 50-experiment budget, while the full system and Autoresearch curves are longer reference trajectories and are not a controlled terminal comparison.

Uses.

- **Direct use:** a research artifact for short budget LM training optimization. The model is below the scale at which broad linguistic competence emerges and is not intended for production language modeling.
- **Out-of-scope use:** downstream language tasks, bias sensitive applications, and any deployment requiring distributional coverage beyond the FineWeb training shard.

Bias, risks, and limitations. The model is trained on FineWeb and inherits any biases of that source. The 300 s training budget produces an underfit model relative to publication scale LMs, so `val_bpb` should not be interpreted as a language task performance score.

Training data. A FineWeb shard distributed with the Autoresearch reference, tokenized by the upstream BPE-8192 tokenizer. The training and evaluation data specifications are inherited verbatim from the task definition shared by all agents.

Training procedure.

- **Architecture knobs:** `DEPTH` $= 7$, `ASPECT_RATIO` $= 96$ (`model_dim` $= 768$), `HEAD_DIM` $=$ 128 (6 heads), `WINDOW_PATTERN` $=$ `SSSL` with short window ratio $1/8$, RoPE, ResFormer alternating value embeddings.
- **Optimizer:** hybrid Muon (NorMuon variant, Polar-Express normalization at constant 1.0 with 4 Newton–Schulz iterations) for 2D matrix parameters, and AdamW for scalars, embeddings, and gates. Per group learning rates are `EMBEDDING_LR` $= 0.6$, `UNEMBEDDING_LR` $=$ 0.004, `MATRIX_LR` $= 0.04$, `SCALAR_LR` $= 0.5$. `WEIGHT_DECAY` $= 0.2$.
- **Schedule:** `TOTAL_BATCH_SIZE` $= 2^{18}$, `DEVICE_BATCH_SIZE` $= 128$, `WARMUP_RATIO` $= 0$, `WARMDOWN_RATIO` $= 0.5$, `FINAL_LR_FRAC` $= 0$.
- **Training regime:** bf16 mixed precision, and `torch.compile` with full graph capture.

Speeds, sizes, times. 300 s wall-clock training (excluding compilation warmup), 1,293 optimizer steps, 339 M tokens processed, 43.5% model FLOPs utilization, and 58 GB peak VRAM.

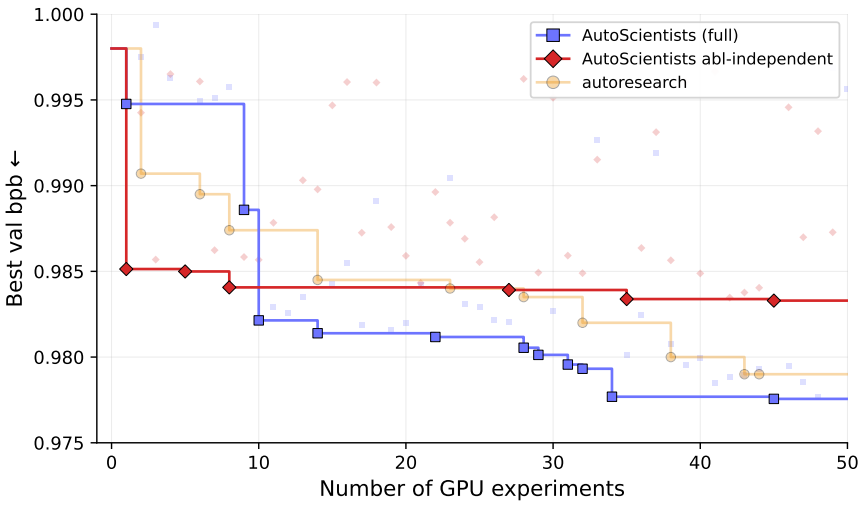


Figure S7: Per experiment trajectory for the `abl-independent` ablation compared with the full AUTOSCIENTISTS system and single-agent autoresearch, all anchored at the pristine nanochat baseline (`val_bpb` $= 0.998$). Solid step curves are the running best-so-far validation `val_bpb` (population minimum across the six solo loops for `abl-independent`), opaque markers indicate global best-so-far drops, and transparent markers show per experiment attempts that did not improve the running min. Final values are autoresearch 0.9773 (15 KEEPs / 83 exps), full AUTOSCIENTISTS 0.9777 (11 KEEPs / 71 exps), `abl-independent` 0.9833 (best-of-6 over 50 exps). The ablation uses a 50-experiment budget, while the full system and Autoresearch curves are longer reference trajectories and are not a controlled terminal comparison.

Evaluation.

- **Test data:** pinned held-out FineWeb shard (about 21 M evaluation tokens), excluded from training.
- **Metric:** validation bits-per-byte, `val_bpb` $=$ total_nats$/(\log 2 \cdot$ total_bytes$)$, with byte length weighting per token.
- **Result:** `val_bpb` $= 0.97769$, reproducible to within the empirical noise floor ($\sigma = 1.0 \times 10^{-3}$, $n = 7$ cross-seed pairs).

Compute infrastructure. Single NVIDIA H100 80 GB GPU. The training script is self-contained, and rerunning it requires only the upstream Autoresearch repository plus the script.

E.2 Research Insights

The companion document recording the reasoning trajectory of the run.

Run details.

- **Task:** GPT nanochat training optimization.
- **Starting metric value:** `val_bpb` $= 0.998$.
- **Final metric value:** `val_bpb` $= 0.97769$.
- **Total experiments run:** 75.
- **Experiments accepted (KEEPs):** 8 (1 baseline lock plus 7 improvements, 10.7% acceptance).
- **Wall-clock duration:** ≈ 6.4 hours.
- **Number of agents:** 9 (6 experiment, 3 analyst).
- **Base LLM:** Claude Sonnet (per agent).

Findings. The seven accepted improvements (Table S7) span three distinct mechanism categories, *throughput* (more optimizer steps within the 300 s budget), *capacity* (more parameters per token), and *optimizer quality* (better gradient signal per step), rather than concentrating on a single research direction. The dominant first improvement is a throughput effect, where halving the per step batch from 2^{19} to 2^{18} approximately doubles the number of optimizer steps at near constant total tokens and alone accounts for -0.0090 of the total -0.017 descent. The largest single capacity step bundles a learning rate correction, since widening the model from $d{=}512$ to $d{=}768$ raises parameter count from 50 M to 94 M and simultaneously brings the d_{model}-LR-scale factor from 0.78 to 1.0. The remaining five wins refine the throughput / quality balance, including `ns_steps` $5 \to 4$ in Polar-Express Muon, two halvings of the short attention window ratio, removing the $1.02\times$ Polar-Express normalization margin, and a final trade of 8% parameters for 14.4% optimizer steps via depth reduction $8 \to 7$. Each refinement is confirmed on multiple seeds against the noise floor.

Table S7: Chain of accepted improvements (KEEPs) produced by AUTOSCIENTISTS on the GPT nanochat task. Each row records the proposing experiment agent, its team, the modified research direction and change applied, the resulting validation `val_bpb`, the Δ from the contemporaneous champion, and the mechanism category recorded by the analyst on the proposing team. Row 0 is the run local baseline lock, and rows 1–7 form the productive descent to the champion at `val_bpb` $= 0.977687$. Mechanism categories are *T* = throughput (more optimizer steps within the budget), *C* = capacity (more parameters per token), *Q* = optimizer quality (better gradient signal per step).

#	Agent	Team	Research direction	Change	val_bpb	Δ	Cat.	Mechanism (analyst recorded)
0	Experiment Agent 3	schedule	—	upstream baseline lock	0.99476	—	—	run local anchor for noise-floor calibration
1	Experiment Agent 1	throughput	TOTAL_BATCH_SIZE	$2^{19} \to 2^{18}$	0.98573	−0.00903	T	$\sim 2\times$ optimizer steps at near constant total tokens
2	Experiment Agent 5	hidden	ASPECT_RATIO	$64 \to 96$ (d_{model} : $512 \to 768$)	0.98214	−0.00359	C	50M→94M params, d_{model} LR scale to 1.0 (removes 22% LR overcorrection)
3	Experiment Agent 6	hidden	ns_steps	$5 \to 4$	0.98106	−0.00108	Q	4 Newton–Schulz iterations sufficient for Polar-Express orthogonalization
4	Experiment Agent 6	hidden	short_window_ratio	$1/2 \to 1/4$	0.98054	−0.00052	T	shorter S-layer window → +2.8% optimizer steps
5	Experiment Agent 6	hidden	short_window_ratio	$1/4 \to 1/8$	0.98013	−0.00042	T	continued bracket of the research direction, +2.2% steps with second seed confirmation
6	Experiment Agent 2	throughput	polar_express_norm	$1.02 \to 1.0$	0.97956	−0.00056	Q	exact normalization in Polar-Express Muon (removes numerical safety margin)
7	Experiment Agent 4	schedule	DEPTH	$8 \to 7$ (at ASPECT_RATIO = 96)	0.97769	−0.00188	T	−8% params (94M→87M) for +14.4% optimizer steps (1130→1293)

Generalizable insights.

1. Under a fixed wall-clock training budget, optimizer step granularity dominates over total token volume on this task, since the same total tokens processed in $\sim 2\times$ as many steps yields a $\sim 1\%$ `val_bpb` reduction. Counterevidence would be a fixed budget benchmark in which step doubling at constant total tokens produces no improvement.
2. Capacity improvements in this regime are gated by accompanying LR scale corrections, since widening the model alone improved `val_bpb` by -0.0036, with the analyst notes attributing most of that to the d_{model}-LR scale becoming exact rather than to raw parameter count. Counterevidence would be a width up KEEP whose internal LR scale factor was already exact.
3. Schedule shape is load-bearing, since pure shape rewrites (e.g. replacing the linear warmdown with a constant) regress, whereas magnitude changes within the existing functional form are productive.

Task-specific findings.

1. Polar-Express Muon's $1.02\times$ numerical stability margin is unnecessary at this scale, since removing it improves `val_bpb` by -0.000564 at 4 Newton–Schulz iterations. Whether this transfers to other Muon implementations or other model scales is not tested by this run.
2. The `SSSL` window pattern with short window ratio $1/8$ is dominant over both pure short (`SSSS`) and shifted (`SSLL`) replacements, and the analyst notes interpret this as preserving a local / long-range attention balance the alternatives break.

Dead ends and negative results. The run rejects 67 experiments, one of which was flagged by the multi seed gate as a near-miss (`short_window_ratio` $= 1/16$, where the two seeds disagreed). Table S8 reports ten representative directions spanning the search space and reveals two different kinds of dead end. The first sets limits on the productive region, as $|\Delta| > 5\sigma$ regressions on `rotary_base` $=$ 1k, `DEPTH` $= 6$, `HEAD_DIM` $= 64$, `ASPECT_RATIO` $= 112$, and `ns_steps` $= 3$ constrain the champion to a narrow ridge in five orthogonal directions. The second characterizes the ridge rather than bounding it, since both `mlp_ratio` directions (3 and 5) are worse than the champion's 4 by similar amounts, all four `WARMDOWN_RATIO` probes $(0.25, 0.30, 0.65, 0.70)$ regress

relative to the champion's 0.5, and replacing the SSSL window pattern with either SSSS or SSLL loses the local / long-range balance.

Table S8: Representative dead ends from the AUTOSCIENTISTS run on GPT nanochat. Of 75 logged experiments, 67 were rejected. Ten representative research directions drawn from across the search space are shown here, and for each, the row reports the direction tested, the value, the resulting `val_bpb`, Δ from the contemporaneous champion (positive by definition for a DISCARD), and the mechanism the analyst recorded for why the change failed.

Axis	Direction	Value	val_bpb	Δ	Reason
DEPTH	decrease	6 (ASPECT_RATIO = 64)	1.0226	+0.0278	capacity collapse from 50M→26M params (−48%) outweighs throughput
HEAD_DIM	decrease	64	0.9904	+0.0094	lower quality attention scores dominate over throughput improvement
ASPECT_RATIO	increase	112	0.9877	+0.0100	per step compute outpaces capacity improvement at fixed budget
TOTAL_BATCH_SIZE	decrease	2^{17}	0.9913	+0.0056	gradient variance penalty exceeds step count improvement
WINDOW_PATTERN	replace	SSSS, SSLL	0.9808, 0.9829	+0.0031, +0.0019	SSSL is load-bearing, and pure short / shifted patterns lose long-range signal
rotary_base	both	1k, 100k	0.9876, 0.9826	+0.0055, +0.0004	frequency band mismatch with 2k-token training context
ns_steps	decrease	3	0.9891	+0.0080	insufficient orthogonalization quality (opposite direction from KEEP at 4)
WARMDOWN_RATIO	both	0.25, 0.30, 0.65, 0.70	0.9856–0.9979	+0.0010 to +0.0099	champion 0.5 sits at an optimum, and both shorter and longer warmdown waste gradient signal
softcap	decrease	30, 10	0.9994, 0.9951	+0.0046, +0.0003	logit softcap clips informative high confidence logits
mlp_ratio	both	3, 5	0.9801, 0.9801	+0.0006, +0.0024	narrow local optimum at champion's value of 4

Coordination and team dynamics. The cold start discussion produced three teams (architecture, schedule, throughput) over 12 substantive forum posts and a $6/6$ DONE / MORE vote split, with no predefined research directions. The accepted improvements distribute across all three teams (4 in architecture, 2 in throughput, and 2 in schedule). Cross-team transfer is observable in the chain, where the short attention window research direction was bracketed by the architecture team, the Polar-Express normalization refinement was proposed by the throughput team on the resulting wider model champion, and the final depth reduction was proposed by the schedule team on top of the throughput team's polar norm KEEP. The multi seed gate prevented one near-miss promotion (`short_window_ratio` $= 1/16$), where the two seeds disagreed.

Limitations of these insights. The findings come from a single run of the system on a single task with a single agent backend at the 300 s budget. The throughput dominance claim is best supported by the -0.009 effect of the first improvement, while the smaller refinement effects (≤ 0.002) are individually multi seed confirmed but their compounding behavior under different orderings is untested, since each refinement was applied to the contemporaneous champion rather than to the pristine baseline. Several research directions were not probed, including pretraining objective changes, weight initialization beyond the embedding init scales tested, and data mixture ratios.

F Implementation Details of BioML-Bench

F.1 Setup

Experiment compute resources. In the BioML-Bench protocol drug discovery, protein engineering, and single cell omics tasks are run on CPU-only machines with an 8-hour limit. In contrast, we run AUTOSCIENTISTS and Autoresearch under a unified experimental compute setting of 4 hours on 1 H100 GPU with 16 CPUs and 48 GB memory for all tasks except biomedical imaging, which had a wall-clock budget of 16 hours. So that every agent receives the same experimental compute, we also rerun MLAgentBench [58], AIDE [11], STELLA [5], and Biomni [10] on the 20 drug discovery, protein engineering, and single-cell omics tasks under this setting (4 hours on 1 H100 GPU with 16 CPUs and 48 GB memory). For these four agents on the four biomedical imaging tasks, we report the published BioML-Bench results, which were obtained with a 16-hour GPU budget. The Reference baseline is a fixed random estimator. We adopt this unified setting, which uses a shorter wall-clock budget, to better reflect contemporary biomedical ML practice, where GPU-backed environments are commonly available even for non-imaging workloads. Additionally, the agents did not use web search or fetch in any of their experiments. The results are available in Table S9.

Validation and test evaluation. For each task, validation sets are defined using domain-specific splitting strategies from the BioML-Bench training set, including scaffold-based holdout splits for drug discovery, patient-level held-out folds for biomedical imaging, and batch or site stratified holdouts for single-cell omics. Final benchmark performance is reported on a held-out BioML-Bench test set for biomedical imaging, drug discovery, and single-cell omics, whereas protein engineering tasks are scored with the benchmark's cross-validation protocol.

F.2 Task Datasets and Evaluation Metrics

F.2.1 Biomedical Imaging

kaggle-histopathologic-cancer-detection. PatchCamelyon (PCam) contains 96×96 RGB pathology image patches with a binary metastatic cancer label defined by the center 32×32 region. Training uses $174{,}464$ labeled patches and the test set has $45{,}561$ patches. The validation set is a random split of the training set. The metric is ROC-AUC (higher is better).

kaggle-osic-pulmonary-fibrosis-progression. A clinical time-series task with 158 idiopathic pulmonary fibrosis patients in the training set (each with multi week FVC measurements and a baseline CT scan). There are 18 test patients with baseline visit only and CT scans. The test grader scores $1{,}908$ `Patient_Week` predictions covering weeks -12 to 133 for the 18 test patients. A patient-level cross-validation split is used for model development. The metric is the modified Laplace log-likelihood $-\sqrt{2}\,\Delta/\sigma_c - \log(\sqrt{2}\,\sigma_c)$ with $\Delta = \min(|FVC_{\text{true}} - FVC_{\text{pred}}|, 1000)$ and $\sigma_c = \max(\sigma, 70)$, averaged over all test (patient, week) pairs (higher is better).

kaggle-rsna-miccai-brain-tumor-radiogenomic-classification. Multi-parametric brain MRI for the RSNA-MICCAI BraTS challenge, where each patient has four DICOM sequences (FLAIR, T1w, T1wCE, T2w). Training uses the 526 patients with binary MGMT promoter methylation labels and the test set is 59 patients. Validation uses a patient-level cross-validation split. The metric is ROC-AUC of the predicted MGMT methylation probability (higher is better).

kaggle-uw-madison-gi-tract-image-segmentation. MR-Linac guided MRI for radiation therapy planning with 50 cancer patients imaged across 1–5 treatment days, with stomach / small bowel / large bowel run-length-encoded segmentation masks. Provided are $77{,}328$ slice class rows in the training set and a $17{,}760$-row validation set with held-out patients and held-out treatment dates. The test set contains $20{,}400$ entries ($6{,}800$ slices $\times$ 3 classes). The metric is $0.4\,\text{Dice} + 0.6\,(1 - \text{HD}_{3D})$, where Dice is per (slice, class) (with 0 for the empty pred / empty truth case) and HD_{3D} is the symmetric Hausdorff distance of the class-OR-merged 3D case day volume, normalized by the unit cube diagonal $\sqrt{3}$ to lie in $[0, 1]$ (higher overall score is better).

F.2.2 Drug Discovery

All drug discovery tasks share the same supervised structure with a public training set and a held-out, label blinded test set, both downloaded from the Polaris Hub. During development the training set is split into five Murcko scaffold groups and 5-fold scaffold cross-validation is used as ℓ_{eval}.

tdcommons-bbb-martins. Binary classification of blood–brain-barrier (BBB) permeability for $1{,}624$ training molecules and 406 test molecules. The metric is ROC-AUC (higher is better).

tdcommons-caco2-wang. Regression of Caco-2 cell permeability ($\log P_{\text{app}}$) for 728 training molecules and 182 test molecules. The metric is mean absolute error (MAE, lower is better).

tdcommons-cyp2d6-substrate-carbonmangels. Binary classification of CYP2D6 substrate activity for 532 training molecules and 135 test molecules. The metric is PR-AUC (higher is better) to remain meaningful under imbalance.

tdcommons-herg. Binary classification of hERG potassium channel inhibition for 523 training molecules and 132 test molecules. The metric is ROC-AUC (higher is better).

tdcommons-lipophilicity-astrazeneca. Regression of octanol/water lipophilicity ($\log D_{7.4}$) for $3{,}360$ training molecules and 840 test molecules from the AstraZeneca DMPK release. The metric is MAE (lower is better).

polaris-adme-fang-hclint-1. Regression of human liver microsome intrinsic clearance (`LOG_HLM_CLint`, mL/min/kg) on the `adme-fang-hclint-1` benchmark with $2{,}229$ training molecules and 575 test molecules. The metric is Pearson correlation r between predicted and measured log clearance (higher is better).

polaris-adme-fang-hppb-1. Regression of human plasma-protein binding (`LOG_HPPB`, % unbound) on the `adme-fang-hppb-1`. There are 126 training molecules and 34 test molecules. The metric is Pearson r (higher is better).

polaris-adme-fang-solu-1. Regression of aqueous solubility (`LOG_SOLUBILITY`) on the `adme-fang-solu-1` benchmark with $1{,}578$ training molecules and 400 test molecules. The metric is Pearson r (higher is better).

polaris-pkis2-egfr-wt-c-1. Binary classification of EGFR wild-type kinase inhibition (`CLASS_EGFR`) on the `pkis2-egfr-wt-c-1` benchmark with 496 training compounds and 144 test compounds and severe class imbalance. The metric is PR-AUC (higher is better) to remain meaningful under imbalance.

F.2.3 Single Cell Omics

open-problems-predict-modality. A bone marrow mononuclear cell (BMMC) CITE-seq dataset in which paired RNA expression ($\sim$13,000 genes) and surface protein abundance ($\sim$134 proteins) are measured in the same cells. Validation uses held-out site/donor batches. The metric is RMSE between predicted and measured protein values across all (cell, protein) pairs (lower is better).

open-problems-single-cell-perturbations. A PBMC perturbation screen with 144 compounds and $5{,}317$ genes, in which differential expression profiles (`clipped_sign_log10_pval` clipped to $[-4, 4]$) are measured per (compound, cell type) pair. Training uses the T-cell, NK-cell and regulatory T-cell rows. Validation is a 20% within cell type split of those training rows. Testing predicts the 151 (compound, cell type) profiles for B cells and Myeloid cells. The metric is Mean Rowwise RMSE (MRRMSE, lower is better).

open-problems-cell-cell-communication-ligand-target. This task contains a triple-negative breast cancer single-cell RNA-seq dataset accompanied by an OmniPath ligand–receptor prior. The supervised labels are sparse, with 81 labeled (ligand, target cell type) pairs and 731 unlabeled test pairs. Validation uses a random $80/20$ split of the 81 labeled pairs. The metric is the odds ratio of true positives in the top-5% scored pairs against the held-out binary response, with the 0.5-shrinkage formula score $= 1 - 1/(1 + \mathrm{OR}/2)$ for boundedness (higher is better).

open-problems-label-projection. A diabetic kidney disease single nucleus RNA-seq dataset with 13 cell types. The validation set is a held-out batch. The metric is the weighted F1-score across cell types (higher is better).

open-problems-spatially-variable-genes. A SlideSeqV2 mouse cortex spatial transcriptomics dataset covering 210 genes and a smaller cerebellum dataset with labels is provided. The task is unsupervised and cerebellum labels may be used to verify that an unsupervised statistic correlates with ground truth. The metric is Kendall's τ between predicted spatial scores and the held-out continuous `spatial_var_score` for the 210 cortex genes (higher is better).

F.2.4 Protein Engineering

ProteinGym datasets do not provide a held-out test set, so the out-of-fold predictions from a prescribed 5-fold cross-validation are themselves the official evaluation, with three split strategies (`fold_random_5`, `fold_modulo_5`, `fold_contiguous_5`) for substitution scans and only `fold_random_5` for indel scans. The final score is the mean Spearman correlation between predicted and measured fitness, computed in raw target space (higher is better). The four substitution mutation DMS datasets are SPIKE_SARS2_Starr_2020_binding, SBI_STAAM_Tsuboyama_2023_2JVG, CBX4_HUMAN_Tsuboyama_2023_2K28, PSAE_PICP2_Tsuboyama_2023_1PSE, and the two indel datasets are Q8EG35_SHEON_Campbell_2022 and CSN4_MOUSE_Tsuboyama_2023_1UFM.

F.3 Performance Across Wall-Clock Time

Here we plot mean Spearman ρ over wall-clock time for AUTOSCIENTISTS vs. Autoresearch on six BioML-Bench Protein Engineering tasks. We draw this comparison for the Protein Engineering subset of BioML-Bench because each task reports the same mean Spearman ρ averaged across the same set of CV splits for ℓ_{eval} and for the final leaderboard performance. Tracking ℓ_{eval} over wall-clock time therefore captures progress on the final leaderboard. In contrast, the other BioML-Bench tasks score a held-out test submission, which evaluates how well the agent searched the model space and how well its final approach generalizes to unseen data. For the Protein Engineering tasks, we show how quickly the champion of each approach improves within the 4h budget (Fig. S8).

Table S9: Performance on four domains in BioML-Bench. All agents use the same experimental compute allocation. MLAgentBench, AIDE, STELLA, and Biomni are rerun under our compute setting on drug discovery, protein engineering, and single-cell omics, and on biomedical imaging, and for Reference, we report the published results from [2]. Detailed results are available in Table S10. Values are mean (SE) across tasks.

Domain	Agent	Leaderboard Percentile (↑)	Mean Rank (↓)	Above Median (% ↑)	Any Medal (% ↑)	Completion Rate (% ↑)
Biomedical Imaging ($n = 4$)	Reference	3.30 (2.52)	–	0.0 (0.0)	0.0 (0.0)	NA
	MLAgentBench°	21.73 (11.41)	4.00	12.5 (12.5)	0.0 (0.0)	100.0
	AIDE°	12.11 (9.43)	4.25	6.2 (6.2)	0.0 (0.0)	81.2
	STELLA°	5.91 (4.98)	4.25	6.2 (6.2)	0.0 (0.0)	68.8
	Biomni°	19.04 (10.83)	4.75	12.5 (12.5)	12.5 (12.5)	100.0
	Autoresearch°	39.60 (21.75)	2.25	25.0 (25.0)	25.0 (25.0)	100.0
	AUTOSCIENTISTS °	**45.75 (22.18)**	**1.50**	**50.0 (28.9)**	**50.0 (28.9)**	100.0
Drug Discovery ($n = 9$)	Reference	1.11 (1.11)	–	0.0 (0.0)	0.0 (0.0)	NA
	MLAgentBench†	37.01 (9.06)	4.33	22.2 (14.7)	22.2 (14.7)	100.0
	AIDE†	42.23 (11.43)	2.78	44.4 (17.6)	33.3 (16.7)	100.0
	STELLA†	16.12 (4.94)	5.44	0.0 (0.0)	0.0 (0.0)	100.0
	Biomni†	47.91 (10.77)	3.11	44.4 (17.6)	44.4 (17.6)	100.0
	Autoresearch†	46.16 (10.59)	2.89	33.3 (16.7)	33.3 (16.7)	100.0
	AUTOSCIENTISTS †	**64.52 (8.37)**	**2.44**	**55.6 (17.6)**	**55.6 (17.6)**	100.0
Protein Engineering ($n = 6$)	Reference	0.00 (0.00)	–	0.0 (0.0)	0.0 (0.0)	NA
	MLAgentBench†	46.97 (20.11)	5.33	50.0 (22.4)	50.0 (22.4)	100.0
	AIDE†	71.21 (11.39)	4.00	66.7 (21.1)	66.7 (21.1)	100.0
	STELLA†	48.48 (19.59)	5.67	50.0 (22.4)	50.0 (22.4)	100.0
	Biomni†	93.94 (3.83)	2.50	**100.0 (0.0)**	**100.0 (0.0)**	100.0
	Autoresearch†	**96.97 (3.03)**	2.00	**100.0 (0.0)**	**100.0 (0.0)**	100.0
	AUTOSCIENTISTS †	**96.97 (3.03)**	**1.50**	**100.0 (0.0)**	**100.0 (0.0)**	100.0
Single Cell Omics ($n = 5$)	Reference	7.34 (4.52)	–	0.0 (0.0)	0.0 (0.0)	NA
	MLAgentBench†	79.00 (10.74)	4.20	80.0 (20.0)	**80.0 (20.0)**	100.0
	AIDE†	58.33 (15.58)	4.40	60.0 (24.5)	60.0 (24.5)	100.0
	STELLA†	74.00 (16.14)	5.20	80.0 (20.0)	60.0 (24.5)	100.0
	Biomni†	78.00 (10.20)	3.60	80.0 (20.0)	**80.0 (20.0)**	100.0
	Autoresearch †	86.00 (9.80)	2.00	**100.0 (0.0)**	**80.0 (20.0)**	100.0
	AUTOSCIENTISTS †	**88.00 (9.70)**	**1.60**	**100.0 (0.0)**	**80.0 (20.0)**	100.0
Overall ($n = 24$)	Reference	2.50 (1.16)	–	0.0 (0.0)	0.0 (0.0)	NA
	MLAgentBench	45.70 (7.43)	4.50	39.6 (10.0)	37.5 (10.1)	100.0
	AIDE	47.81 (7.18)	3.67	46.9 (10.2)	41.7 (10.3)	96.9
	STELLA	34.57 (7.86)	5.25	30.2 (9.4)	25.0 (9.0)	94.8
	Biomni	60.87 (7.23)	3.33	60.4 (10.0)	60.4 (10.0)	100.0
	Autoresearch	66.07 (7.38)	2.38	62.5 (10.1)	58.3 (10.3)	100.0
	AUTOSCIENTISTS	**74.40 (6.20)**	**1.88**	**75.0 (9.0)**	**70.8 (9.5)**	100.0

°† Wall-clock time and compute access of agents are 4h GPU & CPU for † and 16h GPU & CPU for °. All rows use the same compute protocol within each domain, and imaging results for MLAgentBench, AIDE, STELLA, and Biomni are the published values, obtained with a 16-hour GPU budget.

F.4 Behavior of AUTOSCIENTISTS on BioML-Bench Tasks

Here we analyze the approaches taken by AUTOSCIENTISTS. Each task's pipeline is classified into seven non-exclusive method categories, **(1) Gradient-boosted trees** (XGBoost, LightGBM, CatBoost, ExtraTrees, RandomForest); **(2) Foundation model frozen features** (using a pretrained protein/molecule/image model only as a fixed feature extractor); **(3) Foundation model fine-tuning** (LoRA on ESM-2, full ImageNet CNN fine-tuning, end-to-end ChemBERTa); **(4) Custom neural net trained from scratch** (MLP, ResMLP, Chemprop D-MPNN, U-Net without pretraining); **(5) Kernel and instance methods** (RBF-SVM, Tanimoto-SVM, Gaussian process with Tanimoto kernel, SVR, k-NN); **(6) Linear/regularized linear models** (Ridge, RidgeCV, Lasso, LogisticRegression — usually as a meta-learner); **(7) Hand-crafted heuristics** (no learned model).

Biomedical Imaging (4 tasks). Three of four imaging tasks center on ImageNet pretrained CNNs fine-tuned end-to-end. `histopathologic-cancer` fine-tunes a dual stream EfficientNet-B3 [79] with a 3072→512→128→1 fusion head, MixUp, and 4-rotation TTA, `rsna-brain-tumor` fine-tunes EfficientNet-B0 [79] augmented with a slice attention pool and a two-layer Transformer encoder, and `uw-madison-gi` fine-tunes a `segmentation_models_pytorch` U-Net with an EfficientNet-B4 [79] encoder for multi organ segmentation. Foundation model fine-tuning therefore accounts for 3/4 of imaging tasks. The fourth task, `osic-pulmonary-fibrosis`, is distinct, since it uses pre-extracted CNN deep features as a fixed embedding (FM-frozen), fits Ridge regressions for FVC slope and curvature (Linear/Ridge), and retrieves nearest neighbors in PCA-50 CT feature space (Kernel/k-NN). No imaging task uses gradient boosting, a custom NN from scratch, or a heuristic pipeline.

Drug Discovery (9 tasks). Drug discovery pipelines are dominated by gradient-boosted trees on hand-crafted RDKit fingerprint stacks (6/9). The three tasks that omit boosting are `lipophilicity` (ChemBERTa-2 fine-tuned end-to-end), `hclint` (a 10-seed residual MLP ensemble on Morgan + MACCS + AtomPair + Mordred features), and `solu` (Chemprop v2 D-MPNN, 5-fold scaffold CV).

Table S10: Per task comparison of all six agents on BioML-Bench. For each method we report the task score, leaderboard percentile (LB%), whether the score is above the median leaderboard entry (Med), and medal status. LB% is the percentage of leaderboard entries with a lower score and Med compares the score with the leaderboard median, so scores with LB% $= 50.0$ can fall on either side of the median. The best score across the six agents is shown in **bold**. Imaging entries for MLAgentBench, AIDE, STELLA, and Biomni are the published BioML-Bench results over four runs, where LB% counts failed runs as 0, Score averages completed runs (– if none completed), and a decimal in Medal is the fraction of the four runs.

		Biomni				Autoresearch				AUTOSCIENTISTS			
Task	**Metric**	**Score**	**LB%**	**Med**	**Medal**	**Score**	**LB%**	**Med**	**Medal**	**Score**	**LB%**	**Med**	**Medal**
Biomedical imaging													
kaggle-histopathologic-cancer-detection	ROC-AUC	0.81818	20.8	N	–	0.99832	99.3	Y	Gold	**0.99834**	99.3	Y	Gold
kaggle-osic-pulmonary-fibrosis-progression	Laplace LL↑	-9.42451	5.0	N	–	-7.51872	9.8	N	–	**-7.11904**	13.9	N	–
kaggle-rsna-miccai-brain-tumor-radiogenomic-classification	ROC-AUC	0.51338	49.0	0.50	0.50	0.52353	44.3	N	–	0.54353	64.8	Y	Bronze
kaggle-uw-madison-gi-tract-image-segmentation	Dice+HD↑	0.20356	1.4	N	–	**0.56327**	5.0	N	–	0.55114	5.0	N	–
Drug discovery													
tdcommons-bbb-martins	AUROC	0.91014	37.5	N	–	0.90908	25.0	N	–	**0.92030**	75.0	Y	Bronze
tdcommons-caco2-wang	MAE↓	**0.27560**	100.0	Y	Gold	0.32257	25.0	N	–	0.27663	100.0	Y	Gold
tdcommons-cyp2d6-substrate-carbonmangels	AUPRC	0.63100	22.2	N	–	0.67413	22.2	N	–	0.61870	22.2	N	–
tdcommons-herg	AUROC	0.85464	71.4	Y	Silver	0.79985	14.3	N	–	**0.86672**	85.7	Y	Silver
tdcommons-lipophilicity-astrazeneca	MAE↓	0.54186	0.0	N	–	**0.40003**	88.9	Y	Gold	0.42191	77.8	Y	Bronze
polaris-adme-fang-hclint-1	Pearson r	0.71530	60.0	Y	Bronze	**0.73308**	90.0	Y	Gold	0.69122	50.0	N	–
polaris-adme-fang-hppb-1	Pearson r	0.84818	80.0	Y	Silver	0.83560	80.0	Y	Bronze	**0.87292**	80.0	Y	Silver
polaris-adme-fang-solu-1	Pearson r	0.64745	20.0	N	–	0.65073	20.0	N	–	**0.65778**	50.0	N	–
polaris-pkis2-egfr-wt-c-1	PR-AUC	0.74681	40.0	N	–	0.77942	50.0	N	–	0.76616	40.0	N	–
Single Cell Omics													
open-problems-predict-modality	RMSE↓	**0.59122**	100.0	Y	Gold	0.64356	100.0	Y	Gold	0.69722	100.0	Y	Gold
open-problems-single-cell-perturbations	MRRMSE↓	0.78246	80.0	Y	Silver	0.78163	80.0	Y	Silver	**0.77241**	90.0	Y	Gold
open-problems-cell-cell-communication-ligand-target	Odds ratio↑	0.68281	100.0	Y	Gold	0.70835	100.0	Y	Gold	**0.92367**	100.0	Y	Gold
open-problems-label-projection	F1-weighted	0.95437	60.0	Y	Bronze	**0.97484**	100.0	Y	Gold	0.96394	100.0	Y	Gold
open-problems-spatially-variable-genes	Kendall τ	0.64992	50.0	N	–	0.68923	50.0	Y	–	**0.69631**	50.0	Y	–
Protein engineering													
proteingym-dms-SPIKE_SARS2_Starr_2020_binding	Spearman↑	0.59660	81.8	Y	Silver	0.59294	81.8	Y	Silver	**0.67011**	81.8	Y	Silver
proteingym-dms-SBI_STAAM_Tsuboyama_2023_2JVG	Spearman↑	0.80046	81.8	Y	Silver	0.81837	100.0	Y	Gold	**0.83384**	100.0	Y	Gold
proteingym-dms-CBX4_HUMAN_Tsuboyama_2023_2K28	Spearman↑	0.94496	100.0	Y	Gold	0.95137	100.0	Y	Gold	**0.95676**	100.0	Y	Gold
proteingym-dms-PSAE_PICP2_Tsuboyama_2023_1PSE	Spearman↑	0.93362	100.0	Y	Gold	0.90897	100.0	Y	Gold	**0.97606**	100.0	Y	Gold
proteingym-dms-Q8EG35_SHEON_Campbell_2022_indels	Spearman↑	0.80133	100.0	Y	Gold	**0.82683**	100.0	Y	Gold	0.81119	100.0	Y	Gold
proteingym-dms-CSN4_MOUSE_Tsuboyama_2023_1UFM_indels	Spearman↑	0.93493	100.0	Y	Gold	**0.93965**	100.0	Y	Gold	0.93186	100.0	Y	Gold

		MLAgentBench				AIDE				STELLA			
Task	**Metric**	**Score**	**LB%**	**Med**	**Medal**	**Score**	**LB%**	**Med**	**Medal**	**Score**	**LB%**	**Med**	**Medal**
Biomedical imaging													
kaggle-histopathologic-cancer-detection	ROC-AUC	0.87243	26.4	N	–	0.93320	39.7	0.25	–	–	0.0	N	–
kaggle-osic-pulmonary-fibrosis-progression	Laplace LL↑	-8.32835	8.8	N	–	-8.02789	8.7	N	–	-8.43104	2.9	N	–
kaggle-rsna-miccai-brain-tumor-radiogenomic-classification	ROC-AUC	0.53353	51.7	0.50	–	–	0.0	N	–	**0.56118**	20.7	0.25	–
kaggle-uw-madison-gi-tract-image-segmentation	Dice+HD↑	0.00000	0.0	N	–	0.00000	0.0	N	–	–	0.0	N	–
Drug discovery													
tdcommons-bbb-martins	AUROC	0.88575	12.5	N	–	0.90514	25.0	N	–	0.86773	12.5	N	–
tdcommons-caco2-wang	MAE↓	0.30738	50.0	N	–	0.27662	100.0	Y	Gold	0.32976	25.0	N	–
tdcommons-cyp2d6-substrate-carbonmangels	AUPRC	**0.71899**	77.8	Y	Silver	0.66415	22.2	N	–	0.68610	33.3	N	–
tdcommons-herg	AUROC	0.83196	42.9	N	–	0.84433	42.9	N	–	0.78954	14.3	N	–
tdcommons-lipophilicity-astrazeneca	MAE↓	0.54913	0.0	N	–	0.54182	0.0	N	–	0.67589	0.0	N	–
polaris-adme-fang-hclint-1	Pearson r	0.68945	50.0	N	–	0.71598	60.0	Y	Bronze	0.57202	20.0	N	–
polaris-adme-fang-hppb-1	Pearson r	0.81197	60.0	Y	Bronze	0.86624	80.0	Y	Silver	0.52120	0.0	N	–
polaris-adme-fang-solu-1	Pearson r	0.53018	0.0	N	–	0.61415	0.0	N	–	0.55827	0.0	N	–
polaris-pkis2-egfr-wt-c-1	PR-AUC	0.75470	40.0	N	–	**0.78995**	50.0	Y	–	0.71802	40.0	N	–
Single Cell Omics													
open-problems-predict-modality	RMSE↓	0.71235	100.0	Y	Gold	0.79191	20.0	N	–	0.71344	100.0	Y	Gold
open-problems-single-cell-perturbations	MRRMSE↓	0.78233	80.0	Y	Silver	0.77921	80.0	Y	Silver	0.86806	60.0	Y	–
open-problems-cell-cell-communication-ligand-target	Odds ratio↑	0.62828	100.0	Y	Gold	0.65613	100.0	Y	Gold	0.59923	100.0	Y	Gold
open-problems-label-projection	F1-weighted	0.95610	73.3	Y	Bronze	0.95463	66.7	Y	Bronze	0.95825	93.3	Y	Gold
open-problems-spatially-variable-genes	Kendall τ	0.61977	41.7	N	–	0.56144	25.0	N	–	-0.00904	16.7	N	–
Protein engineering													
proteingym-dms-SPIKE_SARS2_Starr_2020_binding	Spearman↑	0.48816	72.7	Y	Bronze	0.58868	81.8	Y	Silver	0.45693	72.7	Y	Bronze
proteingym-dms-SBI_STAAM_Tsuboyama_2023_2JVG	Spearman↑	0.39911	9.1	N	–	0.56342	45.5	N	–	0.41457	18.2	N	–
proteingym-dms-CBX4_HUMAN_Tsuboyama_2023_2K28	Spearman↑	0.90282	100.0	Y	Gold	0.91489	100.0	Y	Gold	0.86738	100.0	Y	Gold
proteingym-dms-PSAE_PICP2_Tsuboyama_2023_1PSE	Spearman↑	0.85182	100.0	Y	Gold	0.89160	100.0	Y	Gold	0.80951	100.0	Y	Gold
proteingym-dms-Q8EG35_SHEON_Campbell_2022_indels	Spearman↑	0.71507	0.0	N	–	0.75969	33.3	N	–	0.75079	0.0	N	–
proteingym-dms-CSN4_MOUSE_Tsuboyama_2023_1UFM_indels	Spearman↑	0.75814	0.0	N	–	0.87693	66.7	Y	Bronze	0.40814	0.0	N	–

Chemistry-specific D-MPNNs (Chemprop) are the primary or co-primary models in two tasks, `solu` and `pkis2-egfr`, and together with the residual MLP that `hclint` uses on fingerprint features they account for the 3/9 NN-scratch tasks. Only `hppb` uses a foundation model in frozen feature mode. The AUTOSCIENTISTS approach for `hppb` appends ChemBERTa-2 embeddings (PCA reduced to 32 dimensions) to a 10-model fingerprint stack before a Ridge meta-learner. `lipophilicity` is the sole task that fine-tunes a foundation model (ChemBERTa-2 via `RobertaForMaskedLM`, CLS-token head). Kernel methods appear in three tasks, where `cyp2d6` adds a calibrated RBF-SVM to its CatBoost+LGB+XGB+ET stack, `hppb` includes three SVR variants (RBF $C \in \{1, 5\}$ and linear), and `pkis2-egfr` uses a Tanimoto-SVM and a Tanimoto-GP alongside four boosting models and a Chemprop MPNN. Linear/Ridge models appear as stack meta-learners in four tasks (`hERG`, `cyp2d6`, `BBB`, `hppb`).

Single Cell Omics (5 tasks). Two of five tasks are hand-crafted, training-free pipelines. `spatially-variable-genes` builds a 24-signal spatial statistics ensemble (Moran's I, Geary's C, SPARK-X, Getis-Ord G^*, variogram, and others) with weight optimization via differential evolution plus coordinate descent, while `cell-cell-communication` scores ligand-receptor pairs using proportion and specificity heuristics with a fine grid search over weights. Two tasks use linear models as their core predictor. `single-cell-perturbations` fits per gene Ridge regressors on four

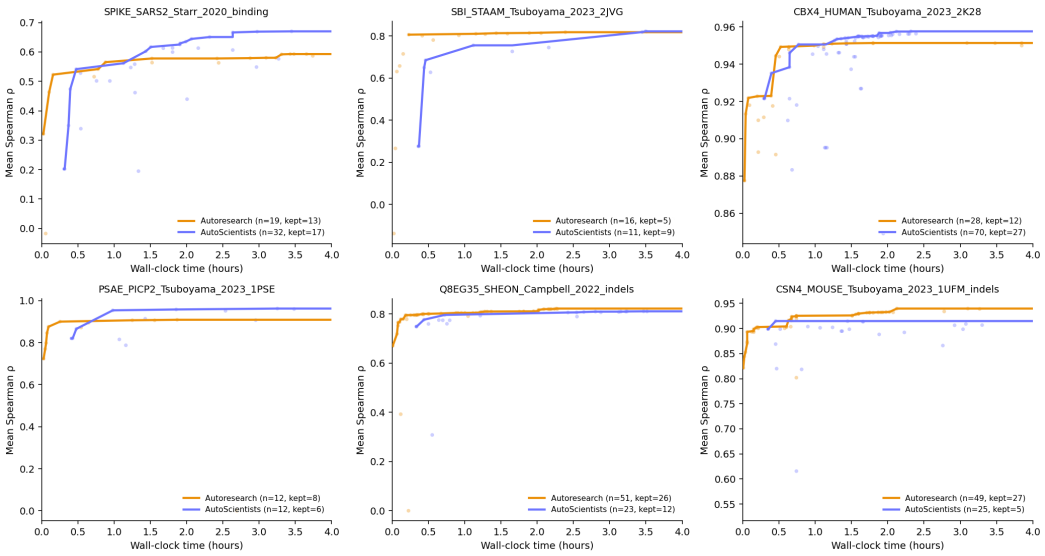


Figure S8: Champion mean Spearman ρ over wall-clock time for AUTOSCIENTISTS vs. Autoresearch on six BioML-Bench Protein Engineering tasks. The solid line tracks the current champion, and the points show experiments that were run but did not outperform the current champion.

interpretable features (T cell, NK cell, Treg expression levels, and Tanimoto drug similarity), tuned via leave-one-out CV, while `label-projection` applies a multinomial LogisticRegression probe to PCA reduced HVG features after batch correction. The remaining task, `predict-modality`, trains a residual MLP (TruncatedSVD input $\rightarrow$ 1024 $\rightarrow$ 512 $\rightarrow$ 256 with skip connections, BatchNorm, and Dropout) from scratch for RNA-to-protein prediction. No single-cell task uses gradient boosting, a pretrained foundation model, or a kernel method.

Protein Engineering (6 tasks). Five of six ProteinGym pipelines rely on frozen ESM-2 embeddings (`SPIKE_SARS2`, `SBI_STAAM`, `Q8EG35`, `CBX4_HUMAN`, `CSN4_MOUSE`), and only `PSAE_PICP2` is built only from fine-tuned models. Two tasks use ESM-2 LoRA fine-tuning as a component, where `CSN4_MOUSE` blends a LoRA model alongside frozen embedding variants, while `PSAE_PICP2` is a pure seven-way LoRA ensemble (ESM-2-35M and ESM-2-8M). The dominant downstream regressor is Ridge or stacked ridge variants (3/6 tasks, `SPIKE_SARS2`, `SBI_STAAM`, `Q8EG35`), followed by gradient boosting as an ensemble member (2/6 tasks, `SPIKE_SARS2` includes a LightGBM submodel, and `Q8EG35` includes XGBoost and LGB submodels). Kernel methods appear only in `Q8EG35`, whose 15-model ensemble includes three SVR variants on ESM-2 features. No ProteinGym pipeline trains a neural net from scratch. Every MLP in the ensembles sits on top of precomputed ESM-2 embeddings, placing them in the FM-frozen category.

Overall trends. Across 24 tasks, Linear/Ridge models are the most frequently occurring category (10/24), but almost always as meta-learners or simple downstream probes on top of richer representations rather than standalone predictors. Gradient-boosted trees on hand-crafted descriptors are the most common *primary* modeling strategy (8/24), concentrated in drug discovery (6/9). Pretrained foundation models are used either as frozen feature extractors (7/24, predominantly in protein engineering) or fine-tuned end-to-end (6/24, predominantly in biomedical imaging and one small molecule task). Kernel methods and custom NNs from scratch each appear in roughly a fifth of tasks (5/24 and 4/24, respectively) and nearly always as components within ensembles. Hand-crafted, training-free pipelines win on the two Single Cell Omics tasks where supervised learning is infeasible due to absence of meaningful labeled training signal. Classical molecular fingerprints (Morgan/MACCS/AtomPair/Torsion/RDKit descriptors) are present in every chemistry pipeline that does not fine-tune a foundation model. Fig. S9 shows the proportion of tasks within each BioML-Bench task family that uses each of the seven method categories.

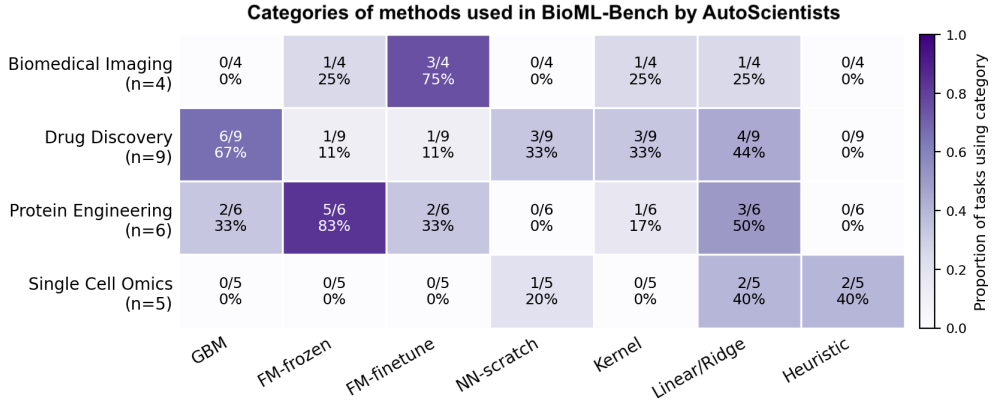


Figure S9: Proportion of tasks within each BioML-Bench task type that uses each method category. The cell value is the fraction of tasks in that row's family in which at least one component of the AUTOSCIENTISTS approach falls in that column's category and categories are non-exclusive (most tasks use several). GBM = gradient-boosted trees; FM-frozen / FM-finetune = pretrained foundation model used as frozen features / fine-tuned; NN-scratch = neural net trained from scratch; Kernel = kernel and instance methods (SVM, SVR, GP, k-NN); Linear/Ridge = Ridge, RidgeCV, Lasso; Heuristic = no learned model.

Table S11: Per task LLM token usage and estimated cost ($ USD) on BioML-Bench. Token counts are reported in millions of tokens (MTok). Costs are estimated using Claude Sonnet 4.6 May 2026 pricing with cache writes charged at the 1-hour cache write price, and exclude GPU compute.

	Autoresearch					AUTOSCIENTISTS				
Task	Input	Output	CacheCreate	CacheRead	Cost ($)	Input	Output	CacheCreate	CacheRead	Cost ($)
kaggle-histopathologic-cancer-detection	0.001	0.4	0.8	100.0	40.1	0.005	0.8	19.6	232.4	199.6
kaggle-osic-pulmonary-fibrosis-progression	0.006	4.8	5.9	535.7	267.9	0.018	5.7	32.8	738.3	503.9
kaggle-rsna-miccai-brain-tumor-radiogenomic-classification	0.002	0.8	1.7	103.4	52.9	0.010	2.9	27.9	491.4	358.0
kaggle-uw-madison-gi-tract-image-segmentation	0.002	1.1	1.7	146.8	70.3	0.008	0.9	10.7	156.1	124.5
open-problems-cell-cell-communication-ligand-target	0.000	0.4	0.6	26.2	17.6	0.005	1.6	8.5	191.3	133.1
open-problems-label-projection	0.000	0.4	0.4	30.9	16.9	0.002	0.6	4.7	79.2	60.5
open-problems-predict-modality	0.000	0.1	0.1	10.2	5.5	0.003	0.7	5.9	165.5	95.0
open-problems-single-cell-perturbations	0.000	0.3	0.4	23.1	13.9	0.007	2.1	13.2	320.0	206.3
open-problems-spatially-variable-genes	0.003	1.8	3.5	219.8	114.0	0.008	2.6	13.4	317.2	214.2
polaris-adme-fang-hclint-1	0.000	0.1	0.2	18.2	8.4	0.007	0.8	7.7	165.6	108.0
polaris-adme-fang-hppb-1	0.000	0.2	0.3	36.7	16.4	0.007	1.7	11.4	419.5	220.4
polaris-adme-fang-solu-1	0.000	0.2	0.2	28.2	12.3	0.003	0.7	5.4	96.9	71.7
polaris-pkis2-egfr-wt-c-1	0.001	0.8	1.1	130.9	58.0	0.007	1.9	14.3	331.9	213.2
proteingym-dms-CBX4_HUMAN_Tsuboyama_2023_2K28	0.001	0.4	0.5	44.2	22.8	0.006	1.5	9.5	286.1	166.2
proteingym-dms-CSN4_MOUSE_Tsuboyama_2023_1UFM_indels	0.001	0.3	0.4	34.4	17.2	0.003	0.7	5.7	106.4	77.5
proteingym-dms-PSAE_PICP2_Tsuboyama_2023_1PSE	0.000	0.1	0.2	9.7	6.0	0.006	1.4	12.1	286.1	179.4
proteingym-dms-Q8EG35_SHEON_Campbell_2022_indels	0.001	1.6	1.7	121.8	70.0	0.005	1.3	10.2	206.2	143.1
proteingym-dms-SBI_STAAM_Tsuboyama_2023_2JVG	0.000	0.3	0.3	19.4	13.0	0.003	0.8	6.7	149.3	96.4
proteingym-dms-SPIKE_SARS2_Starr_2020_binding	0.000	0.6	0.6	25.8	20.6	0.005	1.4	8.9	174.0	126.1
tdcommons-bbb-martins	0.000	0.1	0.2	21.1	9.3	0.009	1.6	12.2	356.2	203.7
tdcommons-caco2-wang	0.000	0.2	0.2	17.7	8.9	0.006	1.3	7.6	398.8	184.2
tdcommons-cyp2d6-substrate-carbonmangels	0.000	0.1	0.2	22.4	9.7	0.014	1.5	9.3	303.1	168.5
tdcommons-herg	0.000	0.2	0.3	33.2	15.2	0.002	0.7	4.5	69.1	57.4
tdcommons-lipophilicity-astrazeneca	0.001	0.4	0.6	47.4	23.2	0.003	0.5	3.9	143.7	73.9
Total	0.024	15.7	21.9	1807.5	910.1	0.153	35.5	266.1	6184.5	3984.7

G AUTOSCIENTISTS-Kermut for ProteinGym SPIKE-SARS2 binding DMS

Our AUTOSCIENTISTS-Kermut champion ProteinGym predictor extends the Kermut Gaussian process model [59] into a three-GP ensemble whose components share a Kermut style structure kernel but differ in their sequence side feature sets and kernel families. All hyperparameters (kernel forms, priors, ensemble composition, region-aware noise schedule, and 1000-step optimization recipe) were chosen on the SPIKE-SARS2 binding task during system development and then frozen. The same configuration is applied unchanged to every other ProteinGym single substitution protein. This champion comes from the first of three independent runs from the Kermut seed, and all three runs and a single-agent Autoresearch control are reported in Appendix C.2.

G.1 Setup

Inputs and per fold preprocessing. For each variant, AUTOSCIENTISTS-Kermut reads (i) a 1280-dimensional ESM-2 `t33_650M_UR50D` mean pool embedding, (ii) the wild-type marginal masked log-probabilities at the mutated position from a ProteinMPNN conditional probability map, used both to form $(\log p_{\text{mut}}, \log p_{\text{wt}}, \log p_{\text{mut}} - \log p_{\text{wt}})$ scalars and as the basis of the structure kernel, (iii) the

ESM-2 zero-shot fitness score, and (iv) fifteen additional zero-shot predictors covering inverse folding, structure-aware, MSA-based and homology-based methods (MIF [81], VenusREM [82], PROSST-128/2048/4096 [83], RSALOR [84], ESCOTT [85], xTrimoPGLM-1B-CLM [86], SaProt-650M-AF2 [87], ESM-IF1 [88], Unirep evotuned [89], S3F-MSA [90], MSA-Transformer ensemble [91], VespaG [92], SiteRM [93]). The four lightweight Cα contact map features appended are the number of residues within 8 Å, the mean residue distance, the local density within 10 Å, and an inverse contact RSA proxy. Within each cross-validation fold, contact features are z-scored using only training statistics, missing extra zero-shot values are filled with the per fold training set median, and the resulting 23-dimensional scalar block is standardized with training fold mean and standard deviation. GP regression targets are quantile normalized to a standard normal on the training fold via van der Waerden scores, where rank r_i (1-indexed, average tied) is mapped to $\Phi^{-1}\big((r_i - 0.5)/N\big)$, keeping quantiles inside $(0, 1)$ and avoiding $\pm\infty$, and test time MSE is computed against z-scored targets to match the ProteinGym/Kermut benchmark convention, with predictions clipped to $[-10, 10]$ to guard against GP extrapolation blow-up on out-of-distribution folds.

Structure kernel. AUTOSCIENTISTS-Kermut adopts the Kermut structure kernel. For each variant indexed by (mutated residue position, mutant amino acid) AUTOSCIENTISTS-Kermut precomputes three pairwise residue matrices on the training fold, (i) a Hellinger distance between the full 20-amino-acid ProteinMPNN conditional probability distributions at the mutated positions, (ii) an L1 distance in log space, $|\log p(\mathrm{aa}_i \mid \mathrm{ctx}_i) - \log p(\mathrm{aa}_j \mid \mathrm{ctx}_j)|$, i.e. the absolute difference of the scalar log-probabilities at the respective mutant amino acids, and (iii) the Euclidean distance between their Cα coordinates. The structure kernel is the radial product $k_{\mathrm{struct}}(i, j) = \exp(-\ell_h h_{ij} - \ell_p p_{ij} - \ell_d d_{ij})$ with positive constrained per component lengthscales, wrapped in a `ScaleKernel` whose output scale carries a $\mathrm{LogNormal}(0, 0.5)$ prior.

Three-GP ensemble. Each ensemble member combines the structure kernel with a sequence kernel via a learnable mixture $k = \pi\, k_{\mathrm{struct}} + (1 - \pi)\, k_{\mathrm{seq}}$, where $\pi = \sigma(\mathrm{raw}_\pi)$ is reparameterized through a sigmoid. The mean function is a single input `LinearMean` over the ESM-2 zero-shot score, providing a strong unsupervised prior on top of which the GP models residuals. The three members differ only in k_{seq} and in the sequence side feature set. **(GP1)** a MATÉRN-3/2 ARD kernel on the $1280 + 23 = 1303$-dimensional concatenation of ESM-2 embedding and standardized scalars, **(GP2)** a MATÉRN-5/2 ARD kernel on the 1280-d ESM-2 embedding alone, providing a smoother sequence kernel that does not see the auxiliary scalars, and **(GP3)** a linear kernel on a compact 18-d feature vector formed by dropping the five trailing extra zero-shot predictors (ProSST-4096, S3F-MSA [90], MSA-Transformer ensemble [91], VespaG, and SiteRM) from the scalar block. The three log-probability scalars, the ESM-2 zero-shot score, the first ten extra zero-shot predictors, and all four contact map features are retained ($23 - 5 = 18$). Predictions from members that converge are averaged, and if a member fails for numerical reasons (e.g. Cholesky errors on a degenerate fold) it is silently dropped and the remaining members are averaged.

Likelihood and region-aware noise. AUTOSCIENTISTS-Kermut uses a `FixedNoiseGaussianLikelihood` with learnable additional homoscedastic noise carrying a $\mathrm{HalfCauchy}(0.3)$ prior. The fixed per sample noise variances are derived from a region-aware schedule that down-weights training points whose CV fold index is far from the held-out fold. With $b_i = |\mathrm{fold}_i - \mathrm{test_fold}|$, a sample weight $w_i = 1 + 0.5\, e^{-b_i}$ is converted to fixed noise $\sigma_i^2 = \mathrm{clip}(\, 0.05\, (w_i^{-1} - \min_j w_j^{-1}),\, 10^{-4}, \infty)$. This places near zero fixed noise on the most reliable points (those adjacent to the test fold) and inflates noise on points distant from the test region, encouraging the GP to fit local structure rather than distant region trends.

Optimization. Each GP's parameters (sequence kernel ARD lengthscales, structure kernel component lengthscales, structure outputscale, π, mean parameters, and additional likelihood noise) are jointly trained for 1000 steps of AdamW with learning rate 10^{-1} decayed cosine style to 10^{-3}, maximizing the exact GP marginal likelihood. The same number of steps and the same optimizer are used for every protein and every CV fold, with random seeds fixed to 2024.

Cross-validation and reporting. AUTOSCIENTISTS-Kermut trains and evaluates on the 5-fold CV columns provided by the ProteinGym single substitutions benchmark (`fold_random_5`, `fold_modulo_5`, or `fold_contiguous_5`), refitting all per fold preprocessing, per fold ensemble members, and per fold target normalization independently for each held-out fold.

G.2 Ablations of AUTOSCIENTISTS-Kermut

We ablate three key design choices that AUTOSCIENTISTS-Kermut introduces to the original Kermut. All runs use the same five-fold cross-validation on the SPIKE-SARS2 binding task dataset. The ablations are as follows. (1) **No ensemble** keeps only GP1 (full ESM-2 + scalar features, Matérn-3/2 + ARD). GP2 (ESM-2 only, Matérn-5/2) and GP3 (compact 18-d, linear) are removed. (2) **No quantile norm targets** trains on simple per fold z-scored targets instead of mapping training targets to a standard normal via rank quantiles. (3) **No extra zero-shot scores** drops the 15 auxiliary zero-shot predictors (MIF, VenusREM, ProSST-{128,2048,4096}, RSALOR, ESCOTT, xTrimoPGLM-1B-CLM, SaProt_650M_AF2, ESM-IF1, Unirep_evotune, S3F_MSA, MSA_Transformer_ensemble, VespaG, SiteRM), and only ESM-2 650M zero-shot remains in the scalar block.

Table S12: Ablations on AUTOSCIENTISTS-Kermut on ProteinGym performance on `SPIKE_SARS2_Starr_2020_binding`. Per split Spearman ↑ and MSE ↓ are reported. The best value is shown in **bold** and the second best is shown in *italics*.

	Contiguous		**Modulo**		**Random**		**Average**	
Variant	ρ	MSE	ρ	MSE	ρ	MSE	ρ	MSE
Kermut	0.6950	0.5830	0.7060	0.5220	0.8410	*0.2070*	0.7473	*0.4373*
AUTOSCIENTISTS	**0.7842**	*0.5229*	**0.8174**	*0.5004*	*0.9204*	0.3347	**0.8407**	0.4527
Ablations								
No ensemble	0.7455	0.6092	*0.8030*	0.5460	**0.9207**	0.3527	*0.8231*	0.5026
No quantile norm	*0.7638*	**0.4595**	0.7782	**0.4263**	0.8791	**0.1645**	0.8070	**0.3501**
No extra ZS	0.7591	0.5802	0.7821	0.5718	0.9109	0.3590	0.8174	0.5037

The ablations in Table S12 show that these three components each contribute to AUTOSCIENTISTS-Kermut's ranking performance. Averaged over the three splits, removing quantile normalized targets causes the largest drop in Spearman correlation (0.8407 to 0.8070, -0.0337), removing the extra zero-shot scores the second largest (to 0.8174, -0.0233), and removing the ensemble the smallest (to 0.8231, -0.0176). The ensemble ablation causes the largest drop only on the contiguous split, because the three GPs carry complementary inductive biases and their diversity is most valuable on this harder split, where sequence extrapolation matters most. Conversely, removing quantile normalization improves MSE, since z-scored targets compress prediction scale. The Spearman drop indicates that mapping skewed DMS scores to a standard normal before fitting the GP improves the learned ranking. The 15 auxiliary zero-shot predictors contribute a consistent improvement in both ρ and MSE across all splits, with no single split driving the effect, suggesting they supply diverse evolutionary signal that the ESM-2 embedding alone does not capture rather than overfitting to a particular data regime.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state that the paper's main methodological contribution is a self-organizing agent team for long-running scientific experimentation, and the claimed empirical improvements are supported by the BioML-Bench, GPT nanochat optimization, and ProteinGym experiments in Sections 4.2 to 4.4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5 discusses limitations including that comparisons match experimental compute but not LLM token usage, accessibility and dependence on closed-source LLM backends, the task dependence of team size, limited evaluation of parallel experimentation under larger GPU budgets, and the need to dynamically scale team size.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not present new theoretical results, theorems, or formal proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides a reproducibility statement, which describes the official repositories and launch setup used for BioML-Bench, ProteinGym, Autoresearch, Biomni, MLAgentBench, AIDE, STELLA, and Kermut, and gives detailed implementation protocols in Appendix A. We also provide a public repository for AUTOSCIENTISTS containing scripts for reproduction. The main experimental settings, datasets, metrics, and compute budgets are further described in Sections 4 and Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code and launch scripts are provided through a public GitHub repository (https://github.com/mims-harvard/AutoScientists) for open access. All data used in this study is from publicly available benchmark repositories and datasets for which we provide details in reproducibility statement on access.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Sections 4.1–4.5 describe the agent backend, baseline comparisons, task settings, and ablations. Additionally Appendices A, F, and G provide the protocol, data splits, evaluation metrics, and task details needed to interpret the experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: The paper reports standard errors for aggregate BioML-Bench and ProteinGym results and includes additional run-to-run stability analyses for GPT nanochat, tdcommons-hERG (with two LLM backends), and both SPIKE-SARS2 settings, together with a per seed noise estimate for the GPT nanochat ablations (Appendices C and B.2). Because repeating all BioML-Bench tasks is computationally expensive, the paper reports variability on representative settings rather than exhaustive multi run replication, and states this scope in Section 5.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper reports the agent backend, H100 GPU usage, CPU and memory settings, wall-clock budgets, and per experiment settings for BioML-Bench, GPT nanochat, and ProteinGym, including the compute protocol used for every baseline (Appendix F) and measured wall-clock time under parallel GPU execution (Appendix B.3). Table S11 further reports compute related details including per task token usage and estimated LLM costs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research uses public benchmark datasets and computational experiments and does not involve human-subject experiments or private user data collection. We have reviewed the NeurIPS Code of Ethics and are not aware of any deviations.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Refer to the paper's impact statement. The statement discusses potential positive impacts from accelerating biomedical and scientific ML experimentation, as well as risks including misuse of autonomous optimization agents, propagation of erroneous experimental conclusions, compute accessibility concerns, and the need for human oversight and controlled release, together with the safeguards that broader deployment would require.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a high-risk pretrained model, image generator, scraped dataset, or similarly high-risk asset requiring special release safeguards. The released assets are code, protocols, and experimental artifacts from scientific ML benchmarks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites and links the existing datasets, codebases, and models used, and Table S1 reports their licenses, versions or commit identifiers where available, and terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The paper releases AUTOSCIENTISTS code at https://github.com/mims-harvard/AutoScientists under an open-source license. The repository documents the agent roles, the shared state, the coordination protocol, and the environment setup, and it includes the launch scripts for all experiments reported in the paper. The paper introduces no new datasets, and all evaluations use existing public benchmarks credited in Table S1.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

Justification: The paper does not involve crowdsourcing, human-subject experiments, or newly collected participant data. All experiments are conducted using automated agents on existing benchmark tasks and public or benchmark provided datasets.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human-subject research, so IRB approval or equivalent review is not applicable. No study participants were recruited or compensated.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are a central component of the method. AUTOSCIENTISTS uses long-running LLM agents, implemented with Claude Code using Claude Sonnet 4.6, and the same backend is used for the autoresearch baseline. We additionally evaluate AUTOSCIENTISTS with an OpenAI Codex backend (Appendix C.2). The paper describes this usage in Sections 3–4 and in the reproducibility statement.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.