SLDBENCH V2: ACTIVE SCALING LAW DISCOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

Scaling laws predict large-model performance from small experiments, but finding a reliable law itself requires costly experiments. SLDBench tests whether language-model agents can discover scaling laws when all experimental results are given upfront. We introduce SLDBench v2 with two extensions. First, we expand the corpus from 8 to 127 scenarios spanning language-model pretraining, downstream performance, and vision–language transfer, and select 30 challenging scenarios on which a reference power-law fit reaches $R^2 \leq 0.8$; they represent at least 1.09×10^{23} FLOPs of source-reported training compute. Second, we introduce *active scaling law discovery*: an agent chooses which experiments to run, pays each one's recorded cost (e.g., training FLOPs), and revises its law as outcomes arrive, and is scored on the same hidden extrapolation targets as in the passive setting. Across 7 frontier agent systems, median R^2 drops from 0.72–0.75 (passive) to 0.24–0.56 (active), and prediction error rises 1.5–4.7×, depending on the system. Agents favor cheap experiments, spending 9–44% of the full-pool cost on average; the systems that spend least fail most often, and cheap purchases leave experimental conditions needed for extrapolation unobserved. Experimental planning thus remains a key open challenge for agents that assist model development in the prevailing AI-for-AI study.

1 INTRODUCTION

Scaling laws guide foundation-model development by predicting how performance changes with model size, training data, and computation (Kaplan et al., 2020; Hoffmann et al., 2022). Their practical value lies in extrapolation: experiments at smaller scales inform decisions about larger training runs, such as how to split compute between model size and data. A reliable law, however, requires experiments of its own. Researchers must choose which configurations to measure and how much compute to spend before the predictions can guide investment. Efficient scaling law discovery therefore depends on both the quality of a law and the cost of the evidence behind it.

Recent work shows that language-model agents can contribute to this process. SLDBench evaluates whether agents can discover scaling laws from completed experiments across 8 scenarios, while SLDAgent searches over law expressions and fitting procedures to improve extrapolation beyond established human-derived laws on the evaluated tasks (Lin et al., 2026). This formulation preserves an essential feature of scientific discovery: the predictive objective is specified, but the mathematical form of the solution is open-ended. The experimental evidence, however, is supplied upfront, and its cost plays no role. Success in this *passive* setting leaves open whether an agent can discover an accurate law when it must also decide which experiments are worth conducting.

Choosing those experiments is closely coupled to choosing the law. Candidate laws can agree on observed small-scale behavior yet diverge sharply at larger scales. Additional inexpensive measurements may leave this ambiguity unresolved, whereas an experiment at a different scale or parameter combination may help distinguish the candidates. The agent's current hypotheses guide which measurements it seeks next, and a new observation may require revising those hypotheses. The agent must therefore reason about what its evidence can establish, which additional evidence would improve extrapolation, and whether that improvement justifies the experimental cost. Neither the number of acquired measurements nor their goodness of fit alone captures this capability.

We introduce SLDBench v2, extending SLDBench in two complementary directions. First, we collect a corpus of 127 scenarios comprising 3,376,938 prediction records from public experimental

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

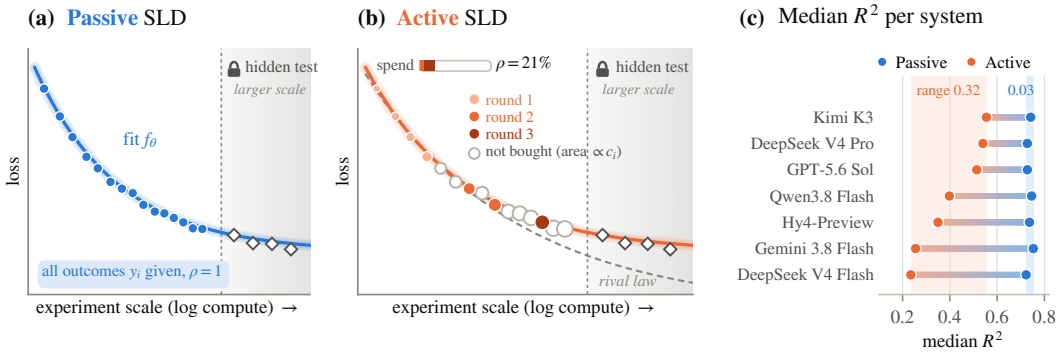

Figure 1: Passive and active scaling law discovery. (a) Passive SLD supplies the outcomes of all candidate experiments; the agent fits a law that must extrapolate to hidden larger-scale runs. (b) Active SLD supplies only a priced catalogue. The agent buys batches of experiments over rounds (filled markers, shaded by round; area grows with price), refits its law after each round, and stops; it is scored on the same hidden runs, and its spending ρ is reported. A rival law (dashed) that fits the cheap points diverges at the test scale. (c) Median R^2 of the 7 agent systems on the 30 tasks: nearly identical in the Passive setting and widely spread in the Active one; shaded bands mark the range of median scores in each setting (Table 1).

data. We construct paired Passive and Active settings for 60 eligible scenarios, then retain the 30 for which the reference power-law fit scores $R^2 \leq 0.8$ in both settings. This screening focuses evaluation on scenarios that remain challenging for a standard law and numerical fitting procedure. The resulting benchmark spans 14 sources, including language-model pretraining, downstream performance, and vision–language transfer, with 339,092 observations. The source-reported training compute of only 2 of these sources already amounts to 1.09×10^{23} FLOPs (Appendix A.4).

Second, we extend the benchmark to *active scaling law discovery* (Active SLD), in which agents allocate experimental resources while constructing a predictive law (Figure 1). An agent receives candidate configurations and their costs, selects experiments, observes their outcomes, and uses the evidence to revise both its law and its next choices. It must also decide when to stop acquiring evidence and submit its law. The form of the law and its fitting procedure remain open-ended. Active SLD thus extends the objective from accurate extrapolation on a fixed dataset to jointly balancing extrapolation quality and experimental cost, such as the FLOPs needed for new measurements.

We implement Active SLD through historical experiment replay: selecting an experiment reveals its recorded outcome and charges its price. This allows decisions about expensive experiments to be evaluated repeatedly without rerunning any training. Each scenario also retains the original passive setting, which provides all candidate outcomes upfront. Both settings use the same hidden extrapolation targets, with predictive quality and experimental cost reported separately.

We evaluate 7 frontier agent systems with 630 runs in each setting (Section 3). Three findings stand out. First, the Passive setting no longer separates these systems, whose median R^2 lies between 0.723 and 0.753, whereas the Active setting spreads them from 0.235 to 0.555, and the Passive ranking does not predict the Active one. Second, the Active loss is a heavier failure tail rather than a uniform shift, and it does not come from sequential access: runs that buy the full pool match their Passive counterparts, so the loss arises from what is bought. Third, agents buy cheap experiments first, the systems that spend least fail most often, and cheap purchases leave groups and parameter combinations unobserved. Non-agent rules point the same way: under a fixed law, sampling each group's price range beats covering every group with its cheapest experiments. By bringing experimental acquisition into scaling law discovery, SLDBench v2 provides a testbed for AI-for-AI research in which agents decide how to spend the compute needed to develop future models.

2 SLDBench v2

This section formalizes passive and active scaling law discovery (SLD), describes how the SLDBench v2 tasks are collected and screened, and specifies the execution environment and metrics. Figure 2 summarizes the construction and follows one task from catalogue to score.

General formulation. Following SLDBench (Lin et al., 2026), an SLD task is a set of experimental records $\mathcal{D} = \{(\mathbf{x}_i, g_i, y_i)\}_{i=1}^{m}$, where each record consists of three components:

(1) a *feature vector* $\mathbf{x}_i \in \mathbb{R}^d$ of experimental choices, such as parameter count N, training tokens or samples D, training compute, learning rate, batch size, or other hyper-parameters;
(2) a *control index* $g_i \in \mathcal{G}$ identifying the context that is held fixed within a single experimental series, such as a model architecture, a training objective, or a family of models;
(3) a *target* $y_i \in \mathbb{R}$, such as a validation loss, an error rate, or a downstream accuracy.

The goal is a symbolic law f with group-specific parameters θ_g such that $\hat{y} = f(\mathbf{x}; \theta_g)$ predicts a hidden test set $\mathcal{T}$ of larger-scale experiments from the same task. The functional form of f is shared across groups, whereas its coefficients may differ. The form itself is not given: for pre-training loss, $L(N, D) = E_g + A_g N^{-\alpha_g} + B_g D^{-\beta_g}$ (Hoffmann et al., 2022) is one hypothesis, and the agent decides which variables enter the law, which parameters are shared across groups, and how the law is fitted. SLDBench supplies the complete set $\mathcal{D}$ upfront; we call this setting *Passive SLD*.

Active formulation. Active SLD withholds the targets and exposes a priced catalogue $\mathcal{U} = \{(i, \mathbf{x}_i, g_i, c_i)\}_{i=1}^{n}$, where $c_i > 0$ is the cost of running experiment i, typically its training FLOPs. The agent starts with $S_0 = \emptyset$. At round t it holds the revealed history $H_t = \{(i, y_i) : i \in S_t\}$ and either stops or buys a new batch $B_t \subseteq \mathcal{U} \setminus S_t$, whose targets are then revealed:

$$S_{t+1} = S_t \cup B_t, \qquad C(S) = \sum_{i \in S} c_i, \qquad \rho(S) = \frac{C(S)}{C(\mathcal{U})} \in [0, 1]. \tag{1}$$

At its stopping round τ the agent submits a law fitted to H_τ. A solution therefore couples three decisions, an acquisition policy, a stopping rule, and a law, and is evaluated by the pair $\big(\mathrm{NMSE}_{\mathcal{T}}(f), \rho(S_\tau)\big)$: how well the law extrapolates, and what fraction of the full-pool cost was spent to obtain it. Passive SLD is the special case $S_0 = \mathcal{U}$ and $\rho = 1$. The coupling is what makes the problem hard. Two candidate laws that agree on the purchased points can diverge at the test scale, and only some purchases, often the expensive ones, can separate them; which purchases are informative in turn depends on the laws still under consideration. Active SLD therefore differs from cost-aware fitting of a supplied law family (Li et al., 2026), where the hypothesis space is fixed, and from pool-based active learning, where test inputs are drawn from the same range as the pool: here the test experiments lie beyond the scale of every candidate and cannot be bought.

Example. Figure 2c shows task T22, from the LAION comparison of CLIP and MaMMUT training (Appendix A). Its features are model size N, samples seen D, and training compute; its target is ADE20K segmentation error, $1 - \mathrm{mIoU}$; and its 22 groups are 11 vision-transformer architectures, each trained with either objective. The catalogue offers 173 runs whose prices span a 15,006-fold range, while the 58 test runs use 0.64–3.07B samples and no candidate exceeds 0.64B. A natural hypothesis is $\hat{y}_g(D) = E_g + A_g D^{-\alpha_g}$, but within a group larger D costs more. The agent must trade breadth against depth: two cheap points in every group cannot identify a per-group exponent, whereas full series for a few groups identify the curve shape only if exponents transfer across groups. Section 3.4 analyzes a run that makes the second choice without testing it.

Datasets. We build SLDBench v2 in three stages (Figure 2a). (i) *Collection.* We gather 127 scenarios with 3,376,938 prediction records from public papers, repositories, and training logs, covering language-model pre-training loss and downstream accuracy, vision and vision–language transfer, multilingual embeddings, optimization hyper-parameters, tokenization, and protein language models. Each scenario records its source, features, target, groups, and extrapolation split; structural curation merges duplicate feature–group keys and removes uninformative features, and provenance and exclusion reasons are retained (Appendix A.1). (ii) *Active eligibility.* A scenario supports a meaningful purchase decision only if every candidate has a positive price, $|\mathcal{U}| \geq 40$, $c_{\max}/c_{\min} \geq 20$, at least 8 distinct prices occur, and, when test prices are recoverable, the median test price exceeds the median candidate price; 60 scenarios qualify. (iii) *Difficulty screen.* We fit a fixed reference, a shared additive power law estimated by bounded multi-start least squares (Appendix A.2), and keep a scenario only if the reference scores $R^2 \leq 0.8$ in both settings; no agent score enters the screen. The result is 30 tasks from 14 sources with 339,092 records, of which 266,183 are candidates and 72,909 are test points. The tasks are heterogeneous (Figure 2b): catalogues contain 40 to 118,480

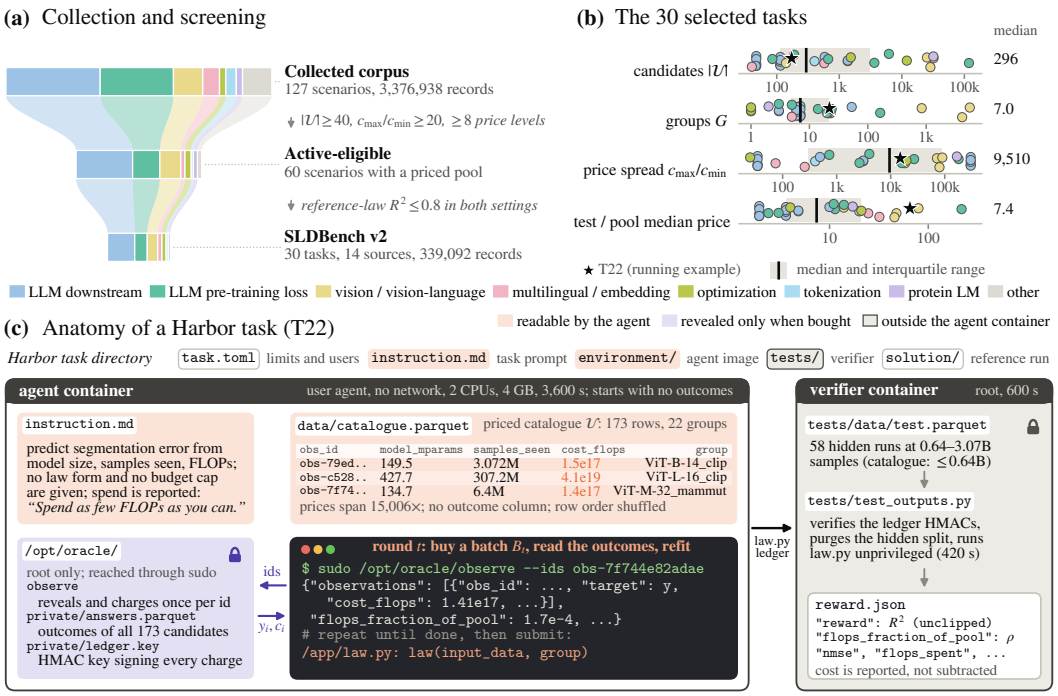

Figure 2: Construction and anatomy of SLDBench v2. (a) The three-stage funnel from the collected corpus to the benchmark, colored by research domain; domains are assigned by keyword rules. (b) The 30 selected tasks. Each dot is a task and the star is the running example T22; the bar and the number on the right give the median, and the band the interquartile range; the last row divides the median test price by the median candidate price. (c) The Harbor task directory and the two containers of one task. The agent reads the instruction and the priced catalogue, buys outcomes through a root-only oracle that signs every charge into a ledger, and submits an executable law; a separate verifier checks the ledger and scores the law on hidden larger-scale runs. File names, commands, and limits are those of the released task.

candidates in 1 to 5,575 groups, the median price spread is 9,510-fold, and in every task the median test experiment costs more than the median candidate (1.9 to 218 times; median 7.4).

Execution environment and metrics. Each task is packaged in the Harbor format (Harbor Framework Team, 2026), the standard task format and evaluation harness of agentic benchmarks such as Terminal-Bench 2.0 (Merrill et al., 2026). A Harbor task is a self-contained directory holding a natural-language instruction, an environment image, a verifier test suite, and a configuration of resources and time limits, so any command-line agent can be evaluated on it without adaptation. A SLDBench v2 task runs as a pair of containers (Figure 2c). The agent container runs as an unprivileged user without network access, with 2 CPUs, 4 GB of memory, and a 3,600 s limit. It holds the instruction and the catalogue, which has no target column. The outcomes reside in a root-only directory reachable only through `sudo /opt/oracle/observe -ids`; the tool returns the targets and prices of the requested candidates, charges each distinct candidate once, and appends an HMAC-signed entry to a spend ledger; repeated purchases are free. The instruction names the target and features, requires one functional form for all groups, gives no law form and no budget cap, and states that spending is reported alongside accuracy. The agent submits `/app/law.py`, which defines `law(input_data, group)`, and a written explanation. A separate verifier container checks the ledger signatures, runs the law as an unprivileged user under a 420 s limit, and scores its predictions on the hidden test set by the normalized mean squared error

$$\mathrm{NMSE}_{\mathcal{T}}(f) = \frac{\sum_{i\in\mathcal{T}}\big(y_i - f(\mathbf{x}_i;\theta_{g_i})\big)^2}{\sum_{i\in\mathcal{T}}(y_i-\bar{y}_{\mathcal{T}})^2}, \qquad R^2 = 1 - \mathrm{NMSE}_{\mathcal{T}}(f). \tag{2}$$

The reward is the unclipped R^2, or zero if the predictions are not finite. $\rho(S_\tau)$ is reported next to it and never subtracted, because any fixed exchange rate between error and FLOPs would encode one practitioner's preference. A negative R^2 means worse than predicting the test mean. The two

settings of a task share one test file, so their NMSE ratio equals their MSE ratio. Replay reveals recorded outcomes instead of launching training jobs, so an episode that buys 10^{20} FLOPs costs only the agent's inference. Prices are source-reported FLOPs, documented estimates such as $6ND$, or non-FLOP proxies (Appendix A.4); agent inference and wall-clock time are not charged.

Specialty of SLDBench v2. Existing formulations fix one of the two unknowns. SLDBench and symbolic regression (Udrescu & Tegmark, 2019; Cranmer, 2023) fix the data and ask for the law; Bayesian experimental design (Huan & Marzouk, 2013) and SL^2 (Li et al., 2026) fix the law family and ask for the data. SLDBench v2 asks for both, while fixing the target, the test set, and the prices and logging every purchase. Error and cost are therefore comparable across systems, and each episode is an auditable record of which experiments an agent chose and in what order (Section 3).

3 Experiments and Analysis

3.1 Experimental setup

Systems and runs. We evaluate 7 agent systems, each a frontier model in a command-line agent harness (Table 1): five models in Claude Code, GPT-5.6 Sol in Codex, and Gemini 3.8 Flash in Gemini CLI. Each system solves every task three times in each setting under the protocol of Section 2, which gives 90 runs per system and setting and 630 runs per setting in total.

Metrics. Quality and cost are reported separately. For quality we report the median R^2 over runs with a finite score and threshold shares, the percentage of all scheduled runs whose R^2 exceeds 0, 0.5, or 0.9, so that a missing score counts as a failure. To compare the settings on a scale-free footing we use the error multiplier $(1 - R^2_{\mathrm{A}})/(1 - R^2_{\mathrm{P}})$ between the Active and Passive runs of the same system, task, and repeat. Because tasks from one source are not independent, we take its median within each of the 14 sources and then across sources, with 95% intervals from 10,000 source resamples (Appendix B.1). Cost is the full-pool cost fraction $\rho(S_\tau)$ of (1).

Baselines. Two kinds of reference place the agents in context. (i) The *reference law* of the difficulty screen, fitted to the full pool. (ii) *Non-agent acquisition rules*: the same reference law fitted to candidates chosen without looking at any outcome, at budgets from 0.5% to 100% of the pool cost (Appendix B.2). *Cheapest-first* buys in ascending price; *random* buys in a random order (three random seeds); *cheapest-per-group* visits the groups in round-robin order and takes each group's next-cheapest candidate; *price-spread* visits the groups in the same way but samples each group's price range coarse to fine, starting from its cheapest and most expensive candidates.

3.2 Main results

Active SLD separates systems that Passive SLD does not. In the Passive setting the 7 systems reach median R^2 between 0.723 and 0.753, a range of 0.030 (Table 1). When the systems must buy their data, the medians span 0.235 to 0.555, a range of 0.320, and the Passive ranking barely predicts the Active one (Spearman $r_s = 0.05$, $n = 7$). Gemini 3.8 Flash has the best Passive median, but the sixth-best Active median and the largest drop, -0.499; Kimi K3, third in the Passive setting, leads in the Active one. Every system loses quality: its source-aggregated error multiplier lies between 1.49 and 4.71, with every 95% interval above one (Figure 3c). The full-pool reference law gives an absolute scale. In the Passive setting every system exceeds it by 0.27 to 0.30; in the Active setting only the three strongest systems do, at median costs of 18–35% of the pool.

The loss is a heavier failure tail. Threshold shares locate the loss (Figure 3a,b). Pooled over systems, the share of runs with $R^2 \leq 0$ or no score rises from 12.5% to 33.0%, the share above 0.5 falls from 71.1% to 45.1%, and the share above 0.9 falls from 18.3% to 7.1% (Table 5). Between 45.6% and 51.1% of the three strongest systems' runs still exceed 0.5, yet 21.1% to 40.0% of each system's Active runs predict worse than the test mean or produce no score.

Sequential access is not the obstacle. An Active run that buys every candidate sees the same data as the Passive run through a different interface. Among the 66 such pairs, Active is better by more

Table 1: Prediction quality and acquisition cost, sorted by median Active R^2. R^2 columns are medians over finite scores; $>.5$ is the percentage of the 90 scheduled runs per system and setting with $R^2 > 0.5$; ΔR^2 is the Active minus the Passive median. Cost is the Active acquisition cost as a percentage of the full-pool cost, and full is the percentage of Active runs that bought every candidate. Bold marks the best agent value of each quality column. †The reference law of the difficulty screen (Appendix A.2), fitted once to all candidates, where the two settings coincide; the screen keeps only tasks on which it scores $R^2 \leq 0.8$.

		Passive		Active			Active cost ρ (%)		
Model	Harness	R^2	$>.5$	R^2	$>.5$	ΔR^2	mean	median	full
Kimi K3	Claude Code	0.742	72.2	**0.555**	45.6	**−0.187**	32.0	17.75	10.0
DeepSeek V4 Pro	Claude Code	0.728	64.4	0.540	**51.1**	−0.188	34.8	21.51	7.8
GPT-5.6 Sol	Codex	0.728	70.0	0.514	**51.1**	−0.214	43.8	35.20	11.1
Qwen3.8 Flash	Claude Code	0.746	74.4	0.398	42.2	−0.348	15.8	5.35	2.2
Hy4-Preview	Claude Code	0.737	**80.0**	0.350	42.2	−0.387	9.1	3.44	0.0
Gemini 3.8 Flash	Gemini CLI	**0.753**	70.0	0.255	42.2	−0.499	25.9	7.78	15.6
DeepSeek V4 Flash	Claude Code	0.723	66.7	0.235	41.1	−0.488	39.3	17.59	26.7
Reference law†	–	0.455	50.0	0.455	50.0	–	100.0	100.00	100.0

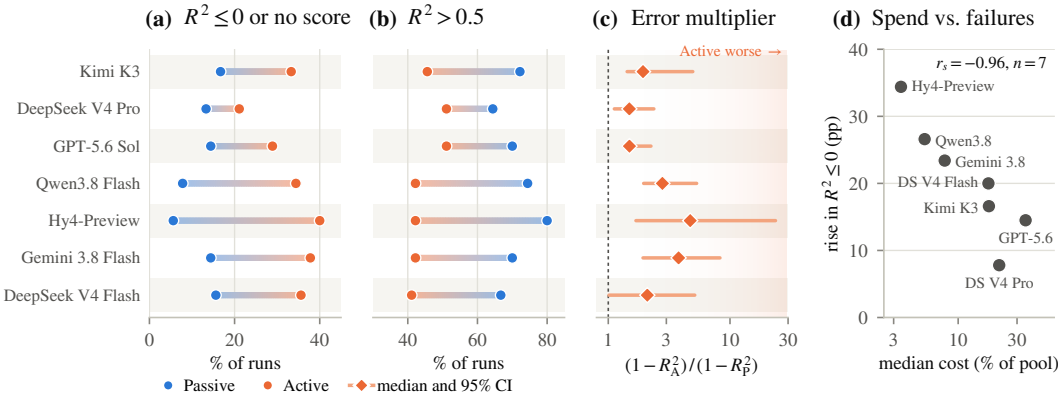

Figure 3: Where the Active loss comes from. (a) Percentage of scheduled runs with $R^2 \leq 0$ or no score and (b) with $R^2 > 0.5$, in the Passive (blue) and Active (orange) settings. (c) Active/Passive error multiplier of each system, aggregated within and then across the 14 sources; bars are 95% source-bootstrap intervals, and the dashed line marks equal error. (d) Median Active cost of each system against the Passive-to-Active rise in its failure share of panel (a); r_s is the Spearman rank correlation over the 7 systems.

than 0.01 in R^2 in 39.4%, Passive in 28.8%, and the remaining 31.8% differ by at most 0.01; the median difference is near zero. These runs are self-selected, but their parity shows that the interface itself imposes no penalty. The Active loss therefore stems from what is bought.

Spending too little predicts failure, but spending is not sufficient. Systems that spend less fail more often (Figure 3d). A system's median cost is almost perfectly rank-anticorrelated with the rise in its failure share ($r_s = -0.96$) and with the fall in its share above 0.5 ($r_s = -0.93$); with $n = 7$ systems, this is a descriptive association. Hy4-Preview and Qwen3.8 Flash, the two lowest spenders at medians of 3.44% and 5.35% of the pool, almost never buy all of it (0% and 2.2% of runs) and raise their failure shares most, by 34.4 and 26.7 percentage points. Spending more does not guarantee success, however. DeepSeek V4 Flash and Kimi K3 have almost the same median cost, 17.59% and 17.75%, yet Active medians of 0.235 and 0.555; DeepSeek V4 Flash spends bimodally, with 22.2% of its runs below 1% of the pool and 27.8% above 90% (Table 8).

3.3 What agents buy

Cheap first, expensive later. Agents prefer cheap candidates. For every system the median share of candidates bought exceeds the median share of cost (Figure 4d), and the median ratio of the mean price paid to the catalogue mean ranges from 0.18 to 0.90 (Table 7); Hy4-Preview buys 40.4% of

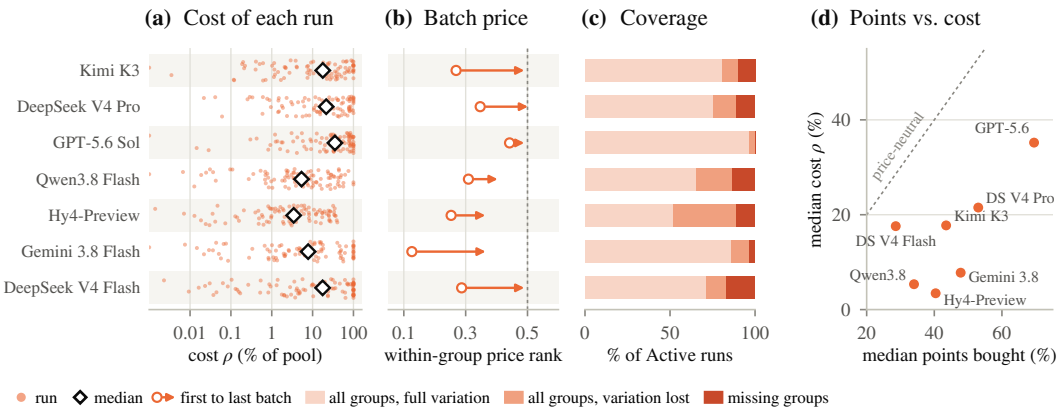

Figure 4: What agents buy. (a) Active cost of every run as a percentage of the full-pool cost (dots) and the per-system median (diamonds). (b) Median within-group price rank of the first (circle) and last (arrowhead) batch of multi-batch runs; 0 is the cheapest candidate of a group and 1 the most expensive. (c) Runs that observe every group with the catalogue's full independent parameter variation, every group with reduced variation, or not every group. (d) Median share of candidates bought against median share of the full-pool cost, each computed separately; points below the diagonal pay less than the catalogue's mean price per candidate.

the candidates for 3.44% of the cost. Within one system, run costs span more than three orders of magnitude (Figure 4a), so a system is better described by a spending distribution than by a budget. The order of purchases follows a common pattern: for every system the median first batch lies in the cheaper half of each group's price range and the median last batch lies higher in that range (Figure 4b). The typical episode is thus cheap exploration followed by costlier confirmation.

Cheap acquisition leaves conditions unobserved. Although 90.6% of Active runs observe every group, 15.2% do so but lose independent parameter variation that the catalogue offers, and 9.4% omit at least one group (Figure 4c). Low spenders are affected most: Hy4-Preview loses variation in 36.7% of its runs and Qwen3.8 Flash in 21.1%, against 3.3% for GPT-5.6 Sol, the highest spender. Such omissions surface at test time. One Hy4-Preview run on OpenWebText buys 51 points but never observes batch size 64 or 128 at learning rate 10^{-4}, both present in the test set, and scores $R^2 = -83.3$; three of the 21 OpenWebText runs miss such a combination (Appendix D).

Outcomes vary across repeats. The same system on the same task often reaches qualitatively different results. Depending on the system, the sign of R^2 changes across the three repeats on 26.7% to 46.7% of tasks, and GPT-5.6 Sol, the highest spender, has a more than tenfold error range on 33.3% of tasks (Table 8). Because acquisition is sequential, an early batch that suggests a wrong law form can plausibly steer the rest of an episode. Per-system medians over three repeats are thus coarse, and differences between adjacent rows of Table 1 should not be over-read.

3.4 What the purchased data are worth

A fixed law needs far more data than the agents buy. We fit the reference law to candidates chosen by the four rules (Figure 5a). Its median R^2 falls steeply with budget, from 0.455 with the full pool to at most 0.36 at $\rho = 50\%$ and at most 0.085 at $\rho = 10\%$, and it is negative under every rule at $\rho \leq 2\%$. The Active medians of the three strongest systems, 0.514–0.555 at median costs of 18–35%, lie above every rule's curve and above the full-pool reference. The comparison conflates acquisition with law choice and favors the agents by construction, since the difficulty screen keeps tasks on which the reference law fails. It shows only that the agents extract more predictive value from partial data than a fixed law does, not that they select their data efficiently.

Spanning each group's price range matters more than covering every group. The rules also show which property of a cheap purchase set matters most (Figure 5c). Random and price-spread selection beat cheapest-first on 63% to 77% of tasks at $\rho \leq 5\%$. Cheapest-per-group, which reaches every group early but only through its cheapest candidates, beats cheapest-first on at most 46.7%

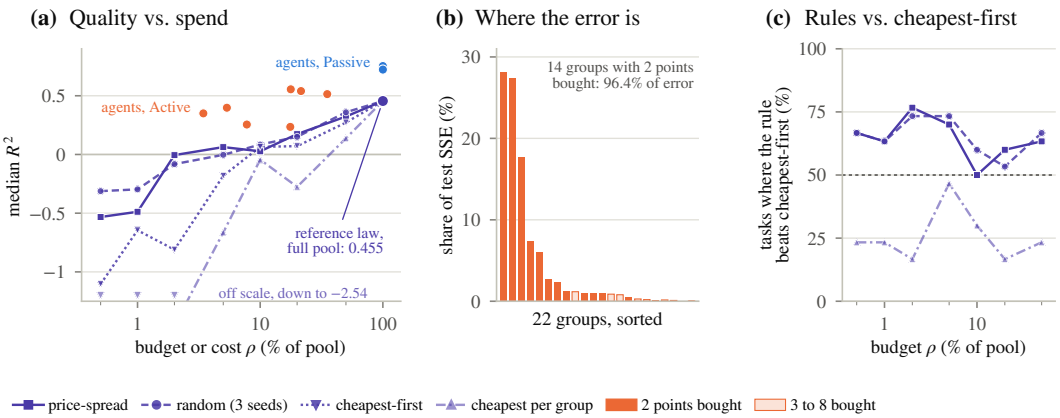

Figure 5: What the purchased data are worth. (a) Median R^2 over the 30 tasks when the fixed reference law is fitted to candidates chosen by four non-agent rules at budget ρ (lines), with the agents' Passive medians at $\rho = 100\%$ and their Active medians at their median cost; each agent marker pairs two separate medians. (b) Share of the test squared error per group in the DeepSeek V4 Pro run on T22, sorted; dark bars are groups with two purchased points. (c) Share of the 30 tasks on which a rule gives the reference law a higher R^2 than cheapest-first selection at the same budget; ties count as not higher, and the dashed line marks parity.

of tasks at any budget, and its median R^2 is the lowest of the four rules at every budget below the full pool. Price-spread visits the groups in the same order and differs only in sampling each group's price range, and at $\rho = 2\%$ it raises the median R^2 from -1.456 (cheapest-per-group) to -0.005. Under a fixed law, cheap points in every group add little information; variation in price within a group, and hence in scale, is what identifies the law, and it is exactly what costs the most.

Error concentrates where an assumption goes untested. Returning to T22, DeepSeek V4 Pro buys eight points in each of four cheap groups to learn the curve shape and two early-scale points in each of 14 other groups, then transfers a shared exponent, $\hat{y}_g = E_g + A_g D^{-0.05}$, to all groups. The run buys 78 points for 4.407% of the pool and scores $R^2 = -10.839$. The 14 sparsely sampled groups contribute 96.4% of the test squared error and the three largest contributors 73.2%, with extrapolation up to 1,000 times the largest purchased sample count (Figure 5b). The failure is not a shortage of points but an assumption the purchases never tested; a few larger-scale points in the sparse groups could have checked whether the exponent transfers. The run thus repeats, with a flexible law, the pattern that makes cheapest-per-group the weakest non-agent rule.

4 Related Work

AI for AI. AI systems already contribute to model development: they search over learning algorithms and update rules (Real et al., 2020; Chen et al., 2023), optimize matrix multiplication and sorting (Fawzi et al., 2022; Mankowitz et al., 2023), evolve executable programs against measured objectives (Romera-Paredes et al., 2024; Novikov et al., 2025), and automate hypothesis generation, experimentation, and manuscript preparation (Lu et al., 2024; Yamada et al., 2025) as well as the assessment of research proposals (Foster et al., 2026). Evaluating a proposal can itself require substantial computation. SLDBench v2 targets the planning of that computation: which measurements an agent should acquire, and at what cost, to predict the behavior of future models.

Research-agent benchmarks. Existing benchmarks evaluate machine-learning experimentation and engineering (Huang et al., 2024; Chan et al., 2025), improvement of research objectives in executable environments (Wijk et al., 2025; Nathani et al., 2025; Garikaparthi et al., 2026; Lupidi et al., 2026), implementation fidelity and reproducibility (Starace et al., 2025; Siegel et al., 2025), and data analysis, hypothesis discovery, and scientific workflows (Chen et al., 2025; Majumder et al., 2025; Bragg et al., 2026). SLDBench v2 makes the experimental evidence itself part of the agent's decision: its paired settings hold the prediction target fixed and vary only access to observations, and purchase records connect extrapolation quality to expenditure and coverage (Appendix E).

Scaling laws and symbolic discovery. Neural scaling laws relate performance to parameters, data, and compute and guide compute-optimal allocation (Kaplan et al., 2020; Hoffmann et al., 2022); later work models transitions between regimes (Caballero et al., 2023) and architectural dimensions beyond parameter count (McLeish et al., 2025). Fitted exponents and extrapolations depend on the observed scale range, the data selected, and the fitting procedure (Choshen et al., 2025; Li et al., 2025), and Active SLD places exactly this dependence under the agent's control. Symbolic regression searches for interpretable expressions that fit measured data (Udrescu & Tegmark, 2019; Cranmer, 2023); SLDBench specializes this search to model-development experiments, and SLDAgent evolves the law expression together with its fitting program (Lin et al., 2026).

Cost-aware experimental design. Bayesian experimental design chooses measurements to reduce uncertainty (Huan & Marzouk, 2013), and Hyperband and BOHB allocate resources across configurations and fidelities (Li et al., 2018; Falkner et al., 2018). Closest to our setting, SL^2 selects experiments from finite pools with heterogeneous costs to fit supplied law families and extrapolate to expensive targets (Li et al., 2026). In SLDBench v2 the agent must also construct the law family, implement the fit, and decide when to stop, with cost reported rather than capped.

5 Limitations and Conclusion

Limitations. Replay reveals recorded outcomes, so an agent can buy only experiments that were run; it cannot propose new configurations, and the catalogue bounds what any policy can learn. Prices mix reported FLOPs, derived estimates, and proxies (Appendix A.4), so ρ compares spending within a task but not absolute compute across tasks. The difficulty screen uses one reference law and thus favors tasks on which it fails. The evaluation covers 7 systems with three repeats on tasks from 14 sources, so per-system estimates are coarse. Finally, cost is reported rather than traded off against quality, so systems are compared on two axes rather than ranked on one.

Conclusion. SLDBench v2 turns scaling law discovery into an experimental-design problem: an agent buys priced experiments from real training records, constructs a law, and is scored on hidden larger-scale runs. Systems that are nearly indistinguishable when all data are given separate widely when they must buy it, and the loss comes from what they buy: agents favor cheap experiments, the lowest spenders fail most often, and cheap purchase sets leave conditions needed for extrapolation unobserved. Non-agent rules indicate that, under a fixed law, variation in scale within each group matters more than reaching every group, and it is also what costs most. Agents that select experiments to discriminate between candidate laws, rather than by price, are a natural next step.

Reproducibility Statement

Section 2 and the appendices specify the task construction, the reference law and difficulty screen, the execution environment and scoring, the statistical procedures, the purchase and coverage diagnostics, and the non-agent baselines. The accompanying scripts reproduce the corpus statistics, the compute accounting, and the non-agent baselines from the task files, and regenerate every experiment figure and table from the recorded evaluation results. Rerunning the agents additionally requires the released Harbor tasks and the models and agent harnesses listed in Table 1.

AI use statement

LLM agent (Codex) was used to polish the writing and figures. The authors verified all text and figures and are fully responsible for the manuscript.

References

Jonathan Bragg, Mike D'Arcy, Nishant Balepur, et al. AstaBench: Rigorous benchmarking of ai agents with a scientific research suite. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.21652v2. 8

Ethan Caballero, Kshitij Gupta, Irina Rish, and David Krueger. Broken neural scaling laws. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.14891. 9

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Madry. MLE-bench: Evaluating machine learning agents on machine learning engineering. *arXiv preprint arXiv:2410.07095*, 2025. URL https://arxiv.org/abs/2410.07095v6. 8

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms. *arXiv preprint arXiv:2302.06675*, 2023. URL https://arxiv.org/abs/2302.06675. 8

Ziru Chen, Shijie Chen, Yuting Ning, et al. ScienceAgentBench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.05080v3. 8

Leshem Choshen, Yang Zhang, and Jacob Andreas. A hitchhiker's guide to scaling law estimation. *arXiv preprint arXiv:2410.11840*, 2025. URL https://arxiv.org/abs/2410.11840v2. 9

Miles Cranmer. Interpretable machine learning for science with PySR and SymbolicRegression.jl. *arXiv preprint arXiv:2305.01582*, 2023. URL https://arxiv.org/abs/2305.01582. 5, 9

Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 1437–1446, 2018. URL https://proceedings.mlr.press/v80/falkner18a.html. 9

Alhussein Fawzi, Matej Balog, Aja Huang, et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610:47–53, 2022. URL https://www.nature.com/articles/s41586-022-05172-4. 8

Thomas Simon Foster, Bassel Al Omari, Tingchen Fu, et al. Ai research preference models. *arXiv preprint arXiv:2608.13940*, 2026. URL https://arxiv.org/abs/2608.13940v1. 8

Aniketh Garikaparthi, Manasi Patwardhan, and Arman Cohan. ResearchGym: Evaluating language model agents on real-world AI research. *arXiv preprint arXiv:2602.15112*, 2026. URL https://arxiv.org/abs/2602.15112. 8

Harbor Framework Team. Harbor: A framework for evaluating and optimizing agents and models in container environments. Software, 2026. URL https://doi.org/10.5281/zenodo.20953922. 4

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022. URL https://arxiv.org/abs/2203.15556. 1, 3, 9

Xun Huan and Youssef M. Marzouk. Simulation-based optimal Bayesian experimental design for nonlinear systems. *Journal of Computational Physics*, 232(1):288–317, 2013. URL https://arxiv.org/abs/1108.4146. 5, 9

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating language agents on machine learning experimentation. *arXiv preprint arXiv:2310.03302*, 2024. URL https://arxiv.org/abs/2310.03302v2. 8

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. URL https://arxiv.org/abs/2001.08361. 1, 9

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018. URL https://www.jmlr.org/papers/v18/16-558.html. 9

Margaret Li, Sneha Kudugunta, and Luke Zettlemoyer. (mis)fitting: A survey of scaling laws. *arXiv preprint arXiv:2502.18969*, 2025. URL https://arxiv.org/abs/2502.18969. 9

Sijie Li, Shanda Li, Haowei Lin, Weiwei Sun, Ameet Talwalkar, and Yiming Yang. Spend less, fit better: Budget-efficient scaling law fitting via active experiment selection. *arXiv preprint arXiv:2604.22753*, 2026. URL https://arxiv.org/abs/2604.22753. 3, 5, 9

Haowei Lin, Haotian Ye, Wenzheng Feng, Quzhe Huang, Yujun Li, Hubert Lim, Zhengrui Li, Xiangyu Wang, Jianzhu Ma, Yitao Liang, and James Zou. Can language models discover scaling laws? In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2507.21184v5. 1, 3, 9, 12

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024. URL https://arxiv.org/abs/2408.06292. 8

Alisia Lupidi, Bhavul Gauri, Thomas Simon Foster, Bassel Al Omari, Despoina Magka, et al. AIRS-Bench: a suite of tasks for frontier AI research science agents. *arXiv preprint arXiv:2602.06855*, 2026. URL https://arxiv.org/abs/2602.06855. 8

Bodhisattwa Prasad Majumder, Harshit Surana, Dhruv Agarwal, Bhavana Dalvi Mishra, Abhijeetsingh Meena, Aryan Prakhar, Tirth Vora, Tushar Khot, Ashish Sabharwal, and Peter Clark. DiscoveryBench: Towards data-driven discovery with large language models. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2407.01725. 8

Daniel J. Mankowitz, Andrea Michi, Anton Zhernov, et al. Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, 618:257–263, 2023. URL https://www.nature.com/articles/s41586-023-06004-9. 8

Sean McLeish, John Kirchenbauer, David Yu Miller, Siddharth Singh, Abhinav Bhatele, Micah Goldblum, Ashwinee Panda, and Tom Goldstein. Gemstones: A model suite for multi-faceted scaling laws. *arXiv preprint arXiv:2502.06857*, 2025. URL https://arxiv.org/abs/2502.06857. 9

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, et al. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*, 2026. URL https://arxiv.org/abs/2601.11868. 4

Deepak Nathani, Lovish Madaan, Nicholas Roberts, et al. MLGym: A new framework and benchmark for advancing AI research agents. In *Conference on Language Modeling*, 2025. URL https://arxiv.org/abs/2502.14499. 8

Alexander Novikov, Ngan Vu, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, et al. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025. URL https://arxiv.org/abs/2506.13131. 8

Esteban Real, Chen Liang, David So, and Quoc Le. AutoML-Zero: Evolving machine learning algorithms from scratch. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 8007–8019, 2020. URL https://proceedings.mlr.press/v119/real20a.html. 8

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, et al. Mathematical discoveries from program search with large language models. *Nature*, 625:468–475, 2024. URL https://www.nature.com/articles/s41586-023-06924-6. 8

Zachary S. Siegel, Sayash Kapoor, Nitya Nadgir, Benedikt Stroebl, and Arvind Narayanan. CORE-Bench: Fostering the credibility of published research through a computational reproducibility agent benchmark. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=BsMMc4MEGS. 8

Giulio Starace, Oliver Jaffe, Dane Sherburn, et al. PaperBench: Evaluating ai's ability to replicate ai research. In *International Conference on Machine Learning*, 2025. URL https://proceedings.mlr.press/v267/starace25a.html. 8

Silviu-Marian Udrescu and Max Tegmark. AI Feynman: A physics-inspired method for symbolic regression. *arXiv preprint arXiv:1905.11481*, 2019. URL https://arxiv.org/abs/1905.11481. 5, 9

Hjalmar Wijk, Tao Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Josh Clymer, Jai Dhyani, Elena Ericheva, Katharyn Garcia, Brian Goodrich, Nikola Jurkovic, Holden Karnofsky, Megan Kinniment, Aron Lajko, Seraphina Nix, Lucas Sato, William Saunders, Maksym Taran, Ben West, and Elizabeth Barnes. RE-Bench: Evaluating frontier AI R&D capabilities of language model agents against human experts. *arXiv preprint arXiv:2411.15114*, 2025. URL https://arxiv.org/abs/2411.15114v2. 8

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025. URL https://arxiv.org/abs/2504.08066. 8

A Task inventory and evaluation protocol

A.1 Task inventory and construction

The selected tasks contain 339,092 data points: 266,183 candidates and 72,909 hidden test points (Table 2). Counts sum prediction records over the 30 tasks and are not multiplied by settings, systems, or repeats; one training run or checkpoint may contribute several records.

Records that share all input features and the group are merged into one record with their median target, so that no two records have identical inputs but different targets. Source-specific metadata defines the scale ordering and the construction of the test set, and Table 3 lists the upstream sources.

A.2 Reference fitting and the difficulty screen

Collection and difficulty selection are separate stages. After structural curation the corpus contains 127 scenarios, of which 60 meet the eligibility conditions of Section 2; these conditions ensure that selecting experiments involves a genuine trade-off between information and cost.

The original SLDBench initializes SLDAgent with a power-law expression and the BFGS optimizer (Lin et al., 2026, Section 3). The reference used for task selection instead fits a shared additive power-law family by bounded, multi-start nonlinear least squares with a soft-L1 penalty on the normalized residuals. Its numerically stabilized form for each group g is

$$f_g(\mathbf{x}) = E_g + \sum_{j=1}^{d} A_{g,j} \exp[-\alpha_{g,j} \log(1 + |x_j|)] \,. \tag{3}$$

Amplitudes and exponents may take either sign, which accommodates both decreasing losses and increasing scores. The fit uses the candidates alone, falls back to a linearized or pooled fit for groups with too few points, and clips predictions to the observed range of the training targets.

Table 2: Data points of the 30 selected tasks, 339,092 in total. Pool and test count the candidate and hidden test records; G counts experimental groups and d input features, as defined in Section 2.

Code	Task	Pool	Test	G	d
T01	IBM BoolQ	47	15	7	2
T02	IBM OpenBookQA	47	15	7	2
T03	IBM OpenBookQA norm.	47	15	7	2
T04	IBM RACE-high	47	15	7	2
T05	IBM WinoGrande	46	15	7	2
T06	OLMo COPA	1,366	342	3	2
T07	OLMo PIQA norm.	1,366	342	3	2
T08	OPT Code lines	114	33	6	2
T09	OPT Human senses	114	33	6	2
T10	OPT Implicatures	114	33	6	2
T11	OPT Swahili proverbs	114	33	6	2
T12	OPT Tao	114	33	6	2
T13	Tokenization	402	101	1	5
T14	Datablations / token–loss	85	29	14	2
T15	DCLM controlled benchmark	583	159	53	2
T16	Gemstones curve	6,213	1,555	3	6
T17	MTEB / param.–performance	575	161	23	1
T18	LAION / compute–downstream error	32,160	8,417	837	2
T19	LAION / param.–downstream error	25,210	9,591	5575	2
T20	LAION / sample–contrastive loss	140	52	21	3
T21	LAION / sample–downstream error	28,632	9,644	4119	2
T22	LAION / sample–segmentation error	173	58	22	3
T23	Loss-to-loss transfer	118,480	29,620	20	4
T24	Misfitting / checkpoint token–loss	3,637	999	163	2
T25	Misfitting / final compute–loss	190	50	5	3
T26	Overtraining loss	664	168	24	3
T27	OWT / batch size–training loss	12,012	3,003	1	3
T28	ProteinLM / compute–loss	31,923	7,982	2	2
T29	Revisiting NMT learning curve	40	13	5	2
T30	StepLaw hyperparam. loss	1,528	383	1	4
	Total	**266,183**	**72,909**	–	–

Table 3: Upstream sources of the selected tasks. Tasks from one source are not independent campaigns.

Source	Upstream project	Tasks
ColPret	IBM/ColPret	12
LAION comparisons	LAION-AI/scaling-laws-for-comparison	5
Misfitting	hadasah/scaling_laws	2
Compute-optimal tokenization	facebookresearch/compute-optimal-tokenization	1
Datablations	huggingface/datablations	1
DCLM	mlfoundations/dclm	1
Gemstones	mcleish7/gemstone-scaling-laws	1
Portuguese MTEB	mteb-pt/mteb-pt-results	1
Loss-to-loss transfer	brendel-group/llm-line	1
Overtraining	mlfoundations/scaling	1
OpenWebText histories	shehper/scaling_laws	1
ProteinLM compute–loss	cxysteven/ScalingProteinLM	1
Revisiting NMT	google-research/revisiting_neural_scaling_laws	1
StepLaw	step-law/steplaw	1

The Active reference buys the entire pool and uses the same implementation. Because numerical fits under a time limit need not coincide, the screen requires both scores to satisfy

$$\max\{R^2_{\text{ref,Passive}},\ R^2_{\text{ref,Active}}\} \le 0.8. \tag{4}$$

This retains 30 scenarios, and no agent score enters the selection. Difficulty is defined relative to this fixed reference law, so the selected tasks are not a representative sample of the corpus.

A.3 Evaluation protocol and acquisition accounting

Both settings of a task use the same hidden test file, and the catalogue order is shuffled. Agents submit `law.py`, `explain.md`, and any required parameter files; the law maps a dictionary of inputs and a group identifier to a prediction. Only the verifier accesses the test targets.

For every task, both versions contain the same multiset of features, groups, and targets; Passive files additionally expose task-specific columns. Twelve tasks derive from ColPret, five from LAION, and two from Misfitting, which motivates the source-level bootstrap of Appendix B.1.

Acquisition cost is the sum of catalogue prices over the distinct purchased candidates, normalized within each task; repeated queries incur no additional charge. Portuguese MTEB is priced by parameter count because its FLOPs are unavailable. The price of a checkpoint may be the cumulative cost of its training run, and the acquisition cost does not deduct computation shared between checkpoints.

A.4 Source-unit compute coverage

Of the selected catalogues, 18 use a stored price column and 12 construct prices as $6ND$. A stored column does not determine the provenance of a cost, since a column named `cost_flops` can contain an estimate or a proxy. We distinguish C3 source-reported FLOPs, C2 FLOPs derived under documented assumptions, C1 non-FLOP proxies, and C0 unavailable costs. In particular, Portuguese MTEB's `cost_flops` equals the parameter count and is therefore C1, whereas OpenWebText's stored training-compute column is derived from $6ND$ and is C2 rather than C3.

To quantify the training computation that the tasks represent, we aggregate over source units, i.e., training runs or configurations, rather than over prediction records, and compute

$$C_{\text{tier}} = \sum_{u \in \mathcal{U}_{\text{tier}}} \max_{r \in u} c_r. \tag{5}$$

Pool and test are joined before grouping, and data shared by the two settings are counted once. Each training trajectory is thus charged once, even if it contributes several checkpoints or metrics. For tokenization, the largest reported budget of each configuration gives a lower bound on its compute without assuming that each reported budget corresponds to an independent training run.

Table 4: Training compute represented by the selected tasks. Units are training runs or configurations, and totals take the maximum recorded cost of each unit over pool and test. C3 and C2 totals are kept separate.

Source	Tier	Accounting unit	Units	FLOPs
Compute-optimal tokenization	C3	Configuration	85	2.70×10^{22}
ProteinLM	C3	Source run	196	8.24×10^{22}
OpenWebText	C2	Source run	15	1.55×10^{18}

The 2 C3 sources alone give the lower bound of 1.09×10^{23} FLOPs stated in the introduction. The C2 estimate in Table 4, all C1 proxies, and sources without a verified mapping from units to costs are excluded, so the bound is neither a corpus total nor the expenditure of an agent.

B Evaluation results and statistical methods

B.1 Score inclusion and aggregation

Medians and error ratios use finite scores. Table 5 reports, for each system and setting, the percentage of all scheduled runs, including unscored ones, whose score exceeds each threshold.

For setting comparisons, the effect is $\log_{10}[(1 - R_A^2)/(1 - R_P^2)]$; positive values favor Passive. An error floor of 10^{-12} handles zero error. Active and Passive runs are paired by system, task, and repeat index. We take the median effect within each of the 14 sources, then report the median across sources with a 95% percentile interval from 10,000 bootstrap resamples of sources.

Table 5: Percentage of all scheduled runs whose R^2 exceeds each threshold.

System	Passive >0	Active >0	Passive $>.5$	Active $>.5$	Passive $>.9$	Active $>.9$
DeepSeek V4 Pro	86.7%	78.9%	64.4%	51.1%	15.6%	11.1%
DeepSeek V4 Flash	84.4%	64.4%	66.7%	41.1%	11.1%	5.6%
GPT-5.6 Sol	85.6%	71.1%	70.0%	51.1%	20.0%	12.2%
Gemini 3.8 Flash	85.6%	62.2%	70.0%	42.2%	18.9%	5.6%
Qwen3.8 Flash	92.2%	65.6%	74.4%	42.2%	20.0%	3.3%
Hy4-Preview	94.4%	60.0%	80.0%	42.2%	23.3%	3.3%
Kimi K3	83.3%	66.7%	72.2%	45.6%	18.9%	8.9%

B.2 Non-agent acquisition baselines

The baselines ask how well the fixed reference law of Appendix A.2 extrapolates from a purchase set chosen without looking at any outcome. For each task and each budget ρ of 0.5, 1, 2, 5, 10, 20, 50, and 100%, a rule scans its candidate order and buys each candidate whose price still fits in the remaining budget $\rho\, C(\mathcal{U})$. Cheapest-first orders candidates by price. Random uses three fixed permutations, and a task's random score is the median over the three seeds, with failed fits ranked worst. Cheapest-per-group orders the groups by their cheapest candidate and visits them round-robin, taking each group's next-cheapest candidate. Price-spread visits the groups in the same order but ranks each group's candidates by price and takes them coarse to fine: cheapest, most expensive, middle, quartiles, and so on. At $\rho = 100\%$ every rule buys the full pool, so it is fitted once. Each fit runs the task's reference implementation on the purchased subset and is scored on the hidden test set with the same unclipped R^2 that the verifier computes for agent submissions.

Table 6 lists each rule's median R^2 and the share of tasks on which it beats cheapest-first at the same budget. Because all rules share one law, differences among them isolate the effect of acquisition, whereas comparisons with the agents (Section 3.4) also reflect law construction.

Table 6: Non-agent acquisition baselines under the fixed reference law. Median R^2 is over the 30 tasks, all of which have finite scores. "Beats cheapest-first" is the percentage of the 30 tasks on which a rule scores a strictly higher R^2 than cheapest-first at the same budget, so ties count against the rule.

	Median R^2				Beats cheapest-first (%)		
ρ (%)	cheapest	random	per group	spread	random	per group	spread
0.5	−1.104	−0.312	−2.541	−0.533	66.7	23.3	66.7
1	−0.646	−0.297	−2.029	−0.488	63.3	23.3	63.3
2	−0.811	−0.082	−1.456	−0.005	73.3	16.7	76.7
5	−0.182	−0.003	−0.664	0.063	73.3	46.7	70.0
10	0.065	0.085	−0.044	0.027	60.0	30.0	50.0
20	0.071	0.150	−0.276	0.173	53.3	16.7	60.0
50	0.272	0.360	0.138	0.323	66.7	23.3	63.3
100	0.455	0.455	0.455	0.455	–	–	–

C Purchase behavior and system-specific patterns

Purchase statistics use runs with at least one purchase, and those involving prediction quality also require a finite score. A full-pool run buys every candidate; a partial run buys a strict subset.

Price preference is measured by the ratio of the mean purchased price to the catalogue mean, which is one for full-pool runs. Batch statistics count newly purchased candidates. In the first and last batches, each candidate receives the midrank of its price within its group, scaled to $[0, 1]$ with lower values cheaper; we average these positions within each batch and report the median over multi-batch runs. Table 7 reports the full-pool share and the medians of point share, cost share, and price ratio, and Table 8 the spread of spending and the stability of outcomes across repeats.

Table 7: Purchase behavior. Full pool is the percentage of runs that buy every candidate; the other columns are separate medians over runs, and those of batch price positions over multi-batch runs (lower is cheaper).

System	Full pool	Point share	Cost share	Price ratio	First-batch price position	Last-batch price position
DeepSeek V4 Pro	7.8%	53.0%	21.51%	0.586	0.347	0.512
DeepSeek V4 Flash	26.7%	28.6%	17.59%	0.904	0.287	0.500
GPT-5.6 Sol	11.1%	69.5%	35.20%	0.716	0.442	0.500
Gemini 3.8 Flash	15.6%	47.8%	7.78%	0.271	0.126	0.375
Qwen3.8 Flash	2.2%	34.0%	5.35%	0.248	0.309	0.414
Hy4-Preview	0.0%	40.4%	3.44%	0.182	0.253	0.374
Kimi K3	10.0%	43.5%	17.75%	0.595	0.269	0.500

Table 8: Spending spread and repeat stability. Costs are percentages of the full-pool cost and the next three columns percentages of Active runs; sign changes of R^2 and more than tenfold error ranges across the three repeats are percentages of the tasks with three finite Active repeats.

System	Cost median (%)	Cost $<1\%$	Cost $>90\%$	Full pool	Sign changes	$>10\times$ error
DeepSeek V4 Pro	21.51	12.2%	8.9%	7.8%	26.7%	23.3%
DeepSeek V4 Flash	17.59	22.2%	27.8%	26.7%	33.3%	23.3%
GPT-5.6 Sol	35.20	7.8%	13.3%	11.1%	43.3%	33.3%
Gemini 3.8 Flash	7.78	21.1%	15.6%	15.6%	46.7%	20.0%
Qwen3.8 Flash	5.35	23.3%	2.2%	2.2%	46.7%	10.0%
Hy4-Preview	3.44	31.1%	0.0%	0.0%	37.0%	11.1%
Kimi K3	17.75	15.6%	13.3%	10.0%	37.5%	16.7%

D COVERAGE AND EXTRAPOLATION CASE STUDIES

Point coverage is the fraction of candidates purchased; group coverage is the fraction of groups with at least one purchase. Experimental groups describe within-task contexts rather than data sources and need not enumerate all parameter combinations. Condition coverage examines parameter values, their combinations, and observed scale ranges through task-specific checks.

Loss of independent parameter variation means that acquired observations distinguish fewer parameter contributions than the full catalogue under the same input representation. Technically, we compare design-matrix rank within each group. Inputs use $\operatorname{sign}(x)\log(1+|x|)$, with a constant column; varying columns and standardization are fixed from the full catalogue. Existing catalogue collinearity is not counted as an acquisition loss, while entirely unobserved groups count as losses. Table 9 separately reports missing groups and runs with reduced independent parameter variation.

Table 9: Experimental coverage, as percentages of all Active runs of each system. Independent variation is measured under the fixed input representation defined above; a run that observes every group can still leave fewer parameter contributions distinguishable than the full catalogue.

System	All groups	Missing groups	Independent variation lost	All groups; variation lost
DeepSeek V4 Pro	88.9%	11.1%	24.4%	13.3%
DeepSeek V4 Flash	83.3%	16.7%	28.9%	12.2%
GPT-5.6 Sol	100.0%	0.0%	3.3%	3.3%
Gemini 3.8 Flash	96.7%	3.3%	14.4%	11.1%
Qwen3.8 Flash	86.7%	13.3%	34.4%	21.1%
Hy4-Preview	88.9%	11.1%	47.8%	36.7%
Kimi K3	90.0%	10.0%	18.9%	8.9%

The OpenWebText example has one explicit group but 15 batch-size and learning-rate combinations. Hy4-Preview acquires 51 observations covering 13 combinations; 30 observations use batch size 2. The combinations of learning rate 10^{-4} with batch size 64 or 128 are never purchased, although both occur in the catalogue and test set; Table 10 lists all such omissions.

Table 10: OpenWebText parameter combinations left unobserved by 14.3% of the runs of this task.

System	Run	Points	Batch	Learning rate	R^2
Hy4-Preview	A	44	4	0.0001	−2.392
Hy4-Preview	A	44	64	0.0001	−2.392
Hy4-Preview	B	51	64	0.0001	−83.305
Hy4-Preview	B	51	128	0.0001	−83.305
Kimi K3	A	135	16	0.0001	−10.580

In the T22 case of Section 3.4, DeepSeek V4 Pro submits $\hat{y}_g(D) = E_g + A_g D^{-0.05}$, with the exponent shared by all groups. It buys all eight candidates of four inexpensive groups, two candidates in each of 14 groups, and three to six in each of the remaining four (Table 11).

Table 11: All 22 groups of the T22 case, sorted by share of the squared test error (SSE); D in millions.

Group	Bought/pool	Bought D	Test D	RMSE	SSE share
ViT-B-16_mammut	2/8	1.28–3.072	640–3072	0.2867	28.16%
ViT-B-32_mammut	2/8	1.28–3.072	640–3072	0.2826	27.35%
ViT-B-16-text-plus_clip	2/8	1.28–3.072	640–3072	0.2273	17.70%
ViT-S-14_mammut	2/8	1.28–3.072	640–3072	0.1469	7.40%
ViT-M-32_mammut	2/8	1.28–3.072	640–3072	0.1319	5.96%
ViT-S-14_clip	2/7	1.28–3.072	1280–3072	0.1083	2.68%
ViT-B-14_clip	2/6	1.28–3.072	1280–3072	0.1005	2.31%
ViT-L-32_mammut	2/8	1.28–3.072	1280–3072	0.0726	1.20%
ViT-S-32_mammut	8/8	1.28–307.2	640–3072	0.0584	1.17%
ViT-S-16_mammut	2/8	1.28–3.072	640–3072	0.0542	1.00%
ViT-M-16_clip	2/8	1.28–3.072	640–3072	0.0536	0.98%
ViT-M-16_mammut	2/8	1.28–3.072	640–3072	0.0522	0.93%
ViT-L-14_mammut	3/8	1.28–6.4	640–3072	0.0504	0.87%
ViT-L-32_clip	5/8	1.28–30.72	640–3072	0.0485	0.80%
ViT-L-16_mammut	2/8	1.28–3.072	1280–3072	0.0481	0.53%
ViT-M-32_clip	8/8	1.28–307.2	640–3072	0.0288	0.28%
ViT-S-32_clip	8/8	1.28–307.2	640–3072	0.0239	0.20%
ViT-L-16_clip	2/8	1.28–3.072	640–3072	0.0275	0.17%
ViT-S-16_clip	8/8	1.28–307.2	1280–3072	0.0265	0.16%
ViT-L-14_clip	4/8	1.28–12.8	640–3072	0.0170	0.10%
ViT-B-14_mammut	2/8	1.28–3.072	640–1280	0.0098	0.02%
ViT-B-32_clip	6/8	1.28–64	1280–3072	0.0083	0.02%

E Comparison with research benchmarks

Table 12 summarizes the benchmarks discussed in Section 4. SLDBench evaluates law construction from supplied observations and SL^2 evaluates acquisition under specified law families; SLDBench v2 evaluates acquisition and law construction jointly, under a priced catalogue.

Table 12: Benchmarks discussed in Section 4, summarized by their primary evaluated activity and outcome.

Benchmark	Evaluated activity	Principal outcome
MLAgentBench / MLE-bench	ML experimentation and engineering	Model or competition score
MLGym	Open-ended AI research tasks	Task-specific improvement
RE-Bench	Research engineering	Progress relative to experts
ResearchGym	Rediscovering a withheld method	Gain over supplied baselines
AIRS-Bench	Research without baseline code	Metrics versus published SOTA
PaperBench / CORE-Bench	Paper replication / reproduction	Rubric score / reproduction
ScienceAgentBench	Data-driven scientific programs	Program execution outcomes
DiscoveryBench	Data-driven hypothesis discovery	Hypothesis facets
AstaBench	Scientific research assistance	Suite-level task outcomes
SLDBench	Law discovery from completed runs	Predictive law quality
SL^2	Acquisition for supplied law families	Extrapolation under cost budgets
SLDBench v2 (ours)	Priced acquisition and law discovery	Prediction, cost, and coverage