ITERLAB: TRAINING AGENTS TO EXPERIMENT AND IMPROVE OVER LONG HORIZONS WITH CALIBRATED SYNTHETIC TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Training agents to improve solutions over long horizons requires tasks that make experimentation both necessary and productive. We introduce IterLab, a pipeline combining calibrated synthetic tasks with supervision on complete experimental histories. Prediction problems with known mechanisms expose structural gains; paired references set quality targets, and recovery checks assess whether input transformations preserve useful signal. Agents inspect data, run code, interpret feedback, and revise solutions within a shared budget. Across 300 tasks, structural features improve reference performance on 214. Quality and interaction screening retain 413 demonstrations from 6,000 requested trajectories. Supervised fine-tuning on these and supplemental demonstrations improves both Qwen3.6-27B and Qwen3.6-35B-A3B on DSBench, MLAgentBench, MLE-bench, and SWE-fficiency. MLE-bench medal rates increase by 15.56 and 14.67 percentage points, respectively, while software-optimization performance improves alongside correctness. Task analyses distinguish modeling from input-processing demands and identify which challenges yield demonstrations. IterLab turns measurable improvement opportunities into training experience for extended, feedback-driven experimentation. Code is available at https://anonymous.4open.science/r/anonymous_code2-CCB1.

1 INTRODUCTION

Developing prediction models or optimizing software requires agents to run experiments and choose what to change next (Huang et al., 2024). Each experiment changes the evidence and budget available for later decisions, giving the task its long horizon. MLE-bench (Chan et al., 2025) and SWE-fficiency (Ma et al., 2026a) evaluate such tasks; training requires experience in interpreting feedback and revising solutions (Qiang et al., 2025; Pan et al., 2025; Nathani et al., 2025).

Training for this process requires tasks with room for improvement (Cobbe et al., 2020; Dennis et al., 2020): routine pipelines offer little reason to experiment, while excessive noise removes learnable signal. Poor agent performance can reflect predictive complexity, limited statistical evidence, or obscured features. Task construction therefore needs measurements that distinguish modeling demands from processing loss. Figure 1 illustrates this task-design requirement.

Environment generators provide execution checks and performance-based selection (Zala et al., 2024; Hu et al., 2025; Shi et al., 2026; Qiang et al., 2026). ML-AutoResearch verifies execution and selects trajectories that improve a baseline (Cai et al., 2026); Synthetic Sandbox fits baselines on hidden-rule tasks to establish performance milestones (Zhou et al., 2026). We focus on calibration: evaluating the same fitting procedure with and without known structural features measures structural benefit, while comparing original, delivered, and recovered inputs measures representation loss.

IterLab trains agents on complete histories of experimentation. Calibrated prediction problems combine structural improvement opportunities with recoverable parsing and feature-engineering demands. Agents inspect public inputs, test candidates, and revise solutions. Grading and interaction screening select successful histories for supervised fine-tuning (SFT) (Zeng et al., 2024; Chen et al., 2023; Pan et al., 2025), retaining the decisions and intermediate failures that led to each submission.

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.

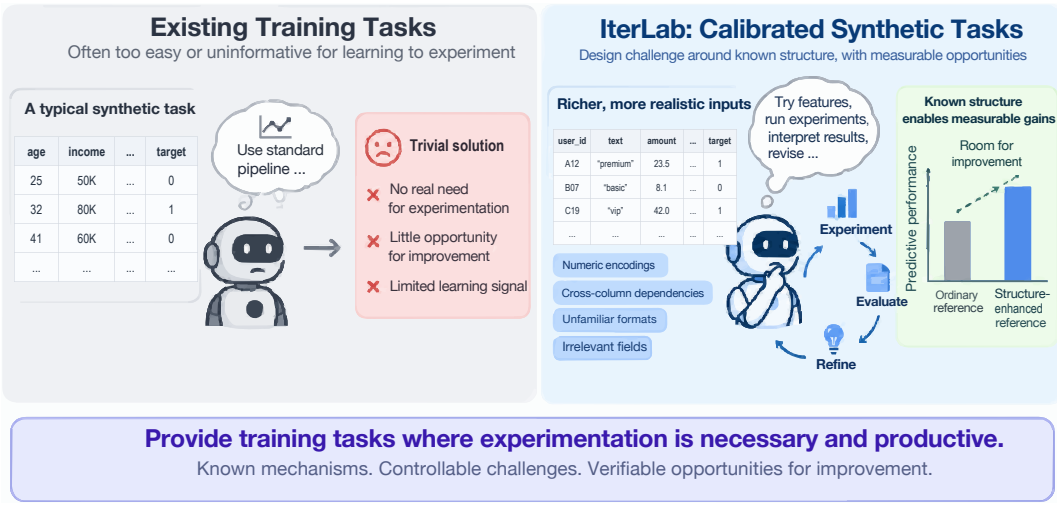


Figure 1: **Why task difficulty needs calibration.** Routine synthetic tasks can offer little need for experimentation. IterLab combines known predictive mechanisms with feature-engineering demands. Paired references quantify improvement opportunities, while recovery checks assess information retention.

Our contributions are calibrated task construction, training from complete experimental histories, and diagnostics linking task demands to available supervision. The 300-task collection yields 413 reviewed demonstrations from 6,000 requested trajectories. Training with these and supplemental demonstrations improves both models on four external benchmarks (Jing et al., 2025; Huang et al., 2024; Chan et al., 2025; Ma et al., 2026a), including MLE-bench gains of 15.56 and 14.67 percentage points. Tasks with positive structural-reference gains dominate generation but supply only 6.2% of automatic candidates. We examine this bottleneck and compare continuous with independent experimentation under matched budgets.

2 Tasks and Agent Interaction

A task specifies public inputs, tools, an execution budget, and a submission contract. Agents receive labeled training data and unlabeled test inputs, construct validation, and implement prediction pipelines. Terminal grading checks validity and private test performance. Mechanisms, reference scores, recovery records, and quality targets remain private. External benchmarks retain their own interfaces and scoring rules (Jing et al., 2025; Huang et al., 2024; Chan et al., 2025; Ma et al., 2026a).

Histories record assistant messages, tool calls, and outputs (Zeng et al., 2024; Chen et al., 2023; Ma et al., 2026b). Agents use observations of quality, errors, and cost to revise code and plan experiments within a shared budget (Huang et al., 2024; Qiang et al., 2025). Earlier observations and remaining resources shape later choices (Qiang et al., 2025; Zhao et al., 2026). Weights stay fixed within an episode; post-training updates them using selected complete histories to improve unseen-task performance (Zeng et al., 2024; Chen et al., 2023; 2025; Ma et al., 2026b; Pan et al., 2025).

3 Calibrated Task Synthesis

IterLab constructs tasks with measurable improvement opportunities. Stage A calibrates a prediction problem; Stage B transforms its inputs and checks recovery. Agents receive the transformed data, objective, metric, and submission contract and conduct experiments within a fixed budget (Figure 2).

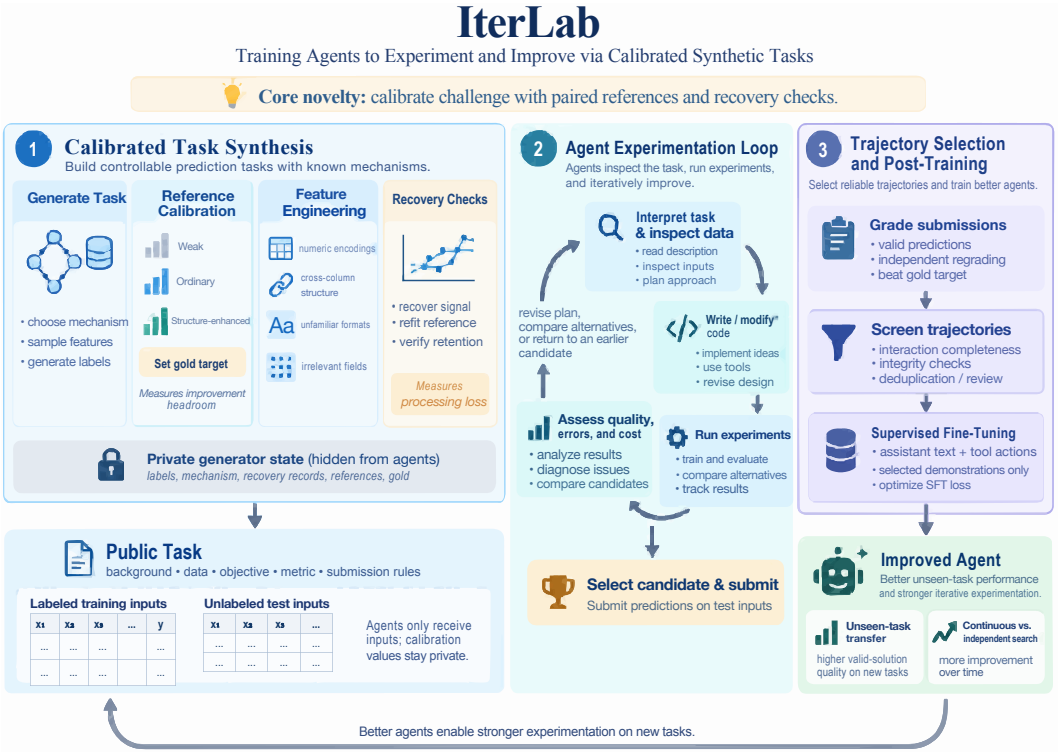


Figure 2: **IterLab framework.** Calibrated synthesis measures signal, structural benefit, and performance retention. Agents experiment with public inputs and feedback; independent grading and screening select SFT histories. Evaluation tests transfer and continuous experimentation.

3.1 Generating prediction problems

We sample a prediction type, mechanism, and statistical parameters. Numeric and categorical features supply a latent signal, converted into class probabilities for label sampling or perturbed by regression noise. Observation noise follows label generation. Mechanisms comprise polynomial, saturating, threshold, parity, hidden-periodic, and rotated-radial relationships. Frequency and interaction order control structure; sample size, noise, redundancy, and class balance control statistical evidence. Appendix A.1 specifies these controls and bounded parameter repair.

We screen usable signal against a weak reference: majority-class prediction for classification, a constant score with AUC 0.5 (Hanley & McNeil, 1982), or the training mean for regression. A lightweight gradient-boosting proxy checks candidate settings (Friedman, 2001). A full predictor portfolio then checks separation from the weak reference relative to score dispersion and rejects excessively easy classification problems. Failed checks trigger bounded parameter relaxations; unresolved configurations are rejected. The admitted task is defined by its final committed parameters.

3.2 Paired references and quality targets

The *ordinary reference* fits the pre-transformation features using a portfolio of linear, tree, kernel, and neural predictors, together with a stacking ensemble (Wolpert, 1992). The *structure-enhanced reference* uses the same fitting procedure with additional features derived from the generating mechanism. For example, a hidden-periodic task supplies the reference with the hidden projection and its sinusoidal features. The comparison measures how much that fitting procedure benefits from structural information; the acting agent receives only the public features.

Let c_w, c_s, and c_o denote the weak, ordinary, and structure-enhanced mean scores. Utility u_m equals the score for AUC and macro-F1 (Hanley & McNeil, 1982; Sokolova & Lapalme, 2009) and its negative for RMSE. Define structural benefit $H = u_m(c_o) - u_m(c_s)$ and weak utility $w = u_m(c_w)$.

Gold is a task-specific quality target:

$$a = \begin{cases} u_m(c_o), & c_o \text{ finite and } H > 0, \\ u_m(c_s), & \text{otherwise,} \end{cases} \qquad u_m(g) = \max\{a - \sigma_a,\ w + 0.3|a - w|\}, \tag{1}$$

where σ_a is the anchor's sample standard deviation. The target follows the stronger fitted reference, with a dispersion adjustment and a minimum improvement over weak. A gold pass requires $u_m(s) > u_m(g)$.

Reference statistics use 40 bootstrap attempts, retaining successful finite replicates (Efron, 1979). Each replicate selects the better test score of a single predictor and a stacking ensemble. Thus gold is a fitted empirical target; test-based selection can raise the reference estimates. Appendix A.2 gives the estimator and admission gates. Execution with public inputs measures the difficulty realized by an agent under its budget.

3.3 Input transformations and recovery

Stage B adds feature-engineering demands while keeping labels and gold fixed. Operators change numeric scales, split or mix columns, merge values into strings, encode dates and categories, introduce recoverable missingness, and add irrelevant fields. Their combinations can make one processing decision depend on another: a merged field must be parsed before its components can be modeled, while a conditional encoding requires examining a second column. Table 5 and Appendix A.3 describe the operators and their associated decisions.

The generator records recovery operations and refits the ordinary reference on both delivered and recovered inputs. Recovered utility must lie no more than two combined standard deviations below the original ordinary reference. This admission gate checks predictive-performance retention under the reference procedure. Measuring both delivered and recovered performance identifies processing loss, while H measures the benefit of additional structural features.

Writing c_n for the delivered-input reference and c_r for the recovered-input reference gives two further diagnostics:

$$V = u_m(c_r) - u_m(c_n), \qquad L = u_m(c_s) - u_m(c_r). \tag{2}$$

Large recovery gain V with small residual loss L identifies transformations that obstruct direct fitting but preserve recovery. Positive H identifies further structural gains. Agents must discover solutions using public inputs.

A language model writes fictional context from predefined fields (Hu et al., 2025; Shi et al., 2026), without data values or private calibration information; a deterministic renderer supplies the schema, metric, and submission requirements. Train and test rows are independently shuffled and identifiers remapped. Appendix B provides the prompts.

4 Learning from experimental histories

We supervise the sequence of decisions behind a successful solution, selecting complete histories by final prediction quality and interaction requirements (Zeng et al., 2024; Chen et al., 2023; Ma et al., 2026b; Chen et al., 2025).

Collection and grading. Each episode supplies a public task, tools, and a fixed execution budget. We record assistant messages, tool calls, and observations through the final response. The agent uses public execution feedback; test grading occurs after submission. Grading checks identifiers and finite predictions, recomputes the task metric, and checks agreement with the recorded score. A valid prediction file can survive an episode timeout, so scoring and eligibility for training are checked separately.

Demonstration selection. A retained history must strictly exceed gold, satisfy normalized-quality and reference-span checks, and complete within budget. Interaction screening requires paired tool calls and results, at least two tool results, and a final assistant response. Suspected private-grader

access and row-order exploitation are reviewed. Exact deduplication compares histories after normalizing incidental identifiers and paths. Intermediate command failures remain when the episode completes and passes selection, preserving subsequent diagnosis and revision. Appendix C specifies the checks.

Supervised fine-tuning. We encode selected histories as complete conversations in the native tool format (Zeng et al., 2024; Chen et al., 2023; Pan et al., 2025; Ma et al., 2026b; Chen et al., 2025). Assistant text and tool actions receive cross-entropy supervision; system, user, and tool-result tokens are masked from the loss. Tool observations remain in the context for predicting subsequent actions. Supervision covers experimental choices, revisions, and recovery from failures in their preceding tool context. Appendix D gives Eq. (9), token accounting, and the training configuration.

5 Experiments

We test transfer from complete experimental histories, characterize task difficulty, and identify which tasks supply supervision. A matched-budget comparison separately measures the benefit of maintaining one continuous history (Zhao et al., 2026).

5.1 Experimental setup

Collection and training. The main collection contains 300 tasks: 100 each for binary classification, four-class classification, and regression. Qwen3.6-27B (Qwen Team, 2026a) and Qwen3.6-35B-A3B (Qwen Team, 2026b) each make ten attempts per task, with 60 CLI turns and 1,800 seconds per episode. Screening retains 413 histories. Full SFT combines these with 269 supplemental histories, allocating 676 to training and six to loss monitoring. Both models train for eight epochs from their corresponding Base checkpoints with seed 42. Appendix D gives the recipe and data composition.

Evaluation. External evaluation uses DSBench (Jing et al., 2025), MLAgentBench (Huang et al., 2024), MLE-bench (Chan et al., 2025), and SWE-fficiency (Ma et al., 2026a), with shared Base/SFT protocols (Table 1). MLAgentBench follows Huang et al. (2024); MLE-bench uses the full competition suite and standard three-run protocol (Chan et al., 2025). SWE-fficiency evaluates one final patch per task (Ma et al., 2026a). Internal evaluation uses 30 tasks excluded from training, ten per prediction type, with two runs for each of the four Base/Full SFT weights: 240 episodes in total. Reported task-bootstrap intervals use 10,000 paired draws after averaging repeats within task (Efron, 1979). The continuous-experimentation study uses 12 of these tasks and 144 episodes under the protocol below. Appendix F specifies evaluation budgets and scoring.

Table 1: **External evaluation scope per model weight.** Repetitions are independent agent episodes per task. Time and action limits apply to each episode. MLE-bench is bounded by wall time; SWE-fficiency evaluates one final patch per episode.

Benchmark	Tasks	Repetitions	Time	Actions
DSBench	74	1	30 min	60
MLAgentBench	13	8	5 h	50
MLE-bench	75	3	24 h	—
SWE-fficiency	498	1	3 h	100

DSBench measures progress between task-specific references (Jing et al., 2025); MLAgentBench counts runs improving the starter metric by at least 10% (Huang et al., 2024); MLE-bench measures bronze-or-higher medal attainment (Chan et al., 2025). SWE-fficiency measures speedup relative to an expert patch and repository-test correctness (Ma et al., 2026a). Each trained weight is compared with its Base checkpoint; rates and gains use counts before rounding (Appendix F).

5.2 Transfer to ML and software-optimization tasks

Consistent gains across external benchmarks. Table 2 shows improvements for both models on all four benchmarks. DSBench RPG rises by 0.0840 for 27B and 0.1609 for 35B-A3B. MLAgent-

Bench success rises by 12.50 and 14.42 percentage points (pp), while MLE-bench medal rates rise by 15.56 and 14.67 pp. Gains span progress, success, and medal metrics. On DSBench, 35B-A3B's larger gain reverses the model ordering: it starts below 27B and finishes above it.

Table 2: **External transfer before and after Full SFT.** RPG measures DSBench progress relative to task baselines and targets. SR is SWE-fficiency's harmonic-mean normalized speedup score. Changes use unrounded values: rates are in pp, RPG and SR in native units.

		DSBench	MLAgentBench	MLE-bench	SWE-fficiency	
Model	Weights	RPG ↑	Success (%) ↑	Medal (%) ↑	SR ↑	Correct (%) ↑
27B	Base	0.3965	75.96	28.44	0.0893	89.76
	Full SFT	**0.4805**	**88.46**	**44.00**	**0.5317**	**92.97**
	Δ	+0.0840	+12.50	+15.56	+0.4424	+3.21
35B-A3B	Base	0.3627	56.73	31.11	0.2841	82.93
	Full SFT	**0.5236**	**71.15**	**45.78**	**0.4562**	**89.16**
	Δ	+0.1609	+14.42	+14.67	+0.1721	+6.22

Transfer beyond prediction modeling. On SWE-fficiency, 27B's SR increases from 0.0893 to 0.5317, while correctness rises from 89.76% to 92.97%. For 35B-A3B, SR increases from 0.2841 to 0.4562 and correctness from 82.93% to 89.16%. Joint gains in speed and correctness extend the benefit to repository-level optimization, which shares IterLab's code-based, budgeted experimentation but requires revising implementations under correctness constraints (Ma et al., 2026a).

External transfer and internal gold success. Internal gold success changes by −1.67 pp for 27B and +1.67 pp for 35B-A3B, with both task-bootstrap intervals including zero (Appendix F.1). External gains are more consistent. The following analyses examine the targets' structural demands and available demonstrations.

5.3 What calibration and recovery measure

Structural benefit raises the quality target. Structure-enhanced references outperform ordinary references on 214/300 tasks (71.3%); gold exceeds the ordinary reference mean on 210/300 (70.0%). Figure 3 shows the largest normalized gains in parity, periodic, and radial mechanisms. Structural features help the reference portfolio most on these families; polynomial, saturating, and threshold tasks provide lower-gap comparisons. Gold adapts to this measured benefit.

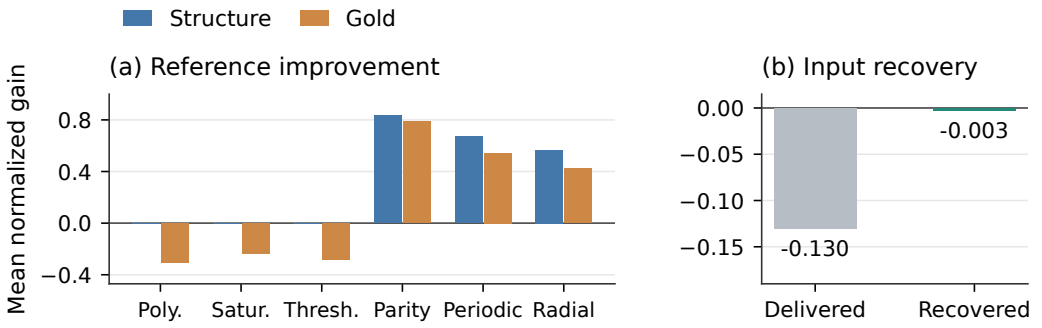


Figure 3: **Calibration separates structural benefit from processing loss.** Differences are normalized by each task's weak-to-structure-enhanced utility span. (a) Structure-enhanced reference and gold relative to the ordinary reference, by mechanism. (b) Ordinary-reference performance on delivered and recovered inputs, relative to the original inputs.

Recovery identifies usable signal behind difficult representations. Recovery improves ordinary-reference performance over delivered inputs on 232/300 tasks (77.3%). Across the col-

lection, the mean normalized difference from the original reference is -0.130 on delivered inputs and -0.003 after recovery; median recovery loss is zero within each metric. Transformations often obstruct direct fitting while preserving recoverable performance. All tasks pass the recovery gate; Appendix A.3 reports task-level variation.

For a hidden-periodic task, ordinary-reference AUC is 0.6991, versus 0.8444 with structural features. Transformations reduce direct-fitting AUC to 0.5053; recovery restores 0.6991, while gold remains 0.8314. These comparisons separate processing demands from improvement beyond ordinary fitting, which the delivered-table score alone obscures (Appendix A.4).

5.4 How success varies across task mechanisms

Mechanism settings and sample coverage. Figure 4 groups all 300 tasks by type, family, and committed parameter, with ten requested episodes per model and task under a 30-minute/60-turn budget. Periodic frequencies $\omega \in \{4, 5, 6\}$ occur in 30, 38, and 44 tasks; parity orders $k \in \{4, 5\}$ occur in 22 and 20 binary tasks. Counts by metric and setting appear in the figure and Appendix F.4. Statistical and representation parameters also vary, so these groups describe collected-corpus difficulty.

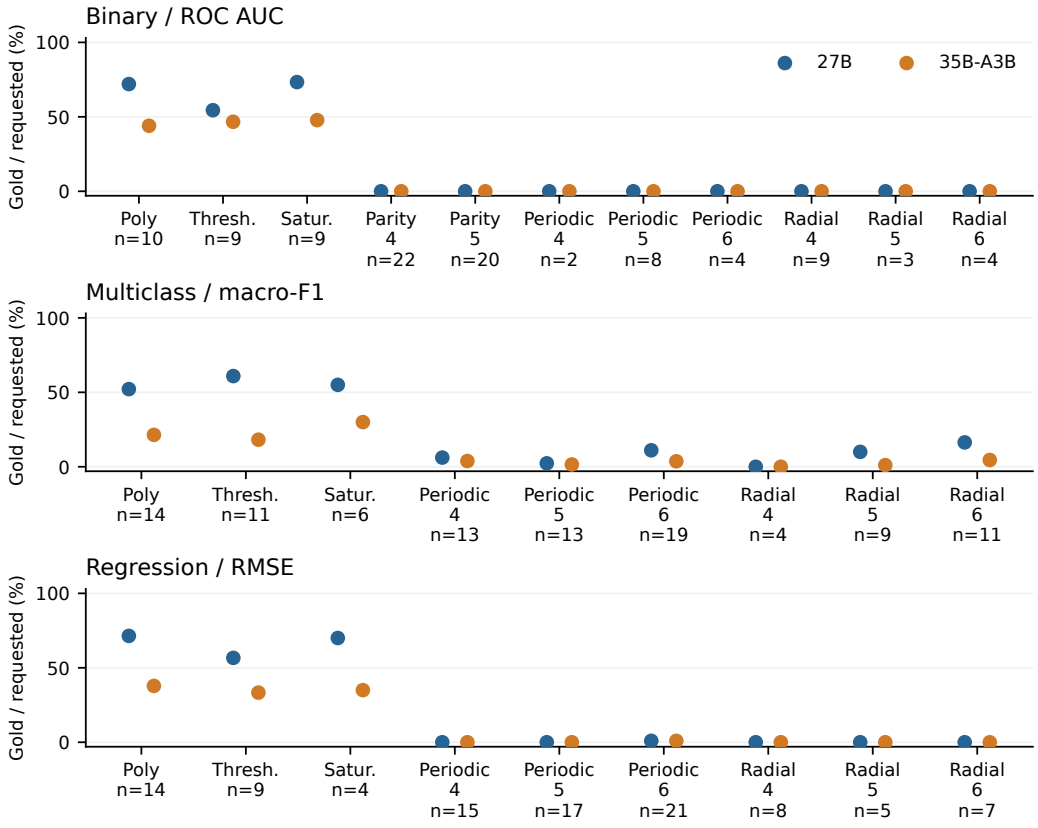


Figure 4: **Agent success across committed mechanism settings.** Each point reports gold passes divided by requested episodes for the indicated model, metric, and setting. Labels give parity order or periodic/radial frequency; n is the number of independent tasks, yielding $10n$ requested episodes per model. Unscored and blocked requests remain in the denominator.

Predictive structure separates reference opportunity from agent success. Both models record zero gold passes across all 42 parity tasks, giving 0/420 requested episodes per model despite the large reference gains in Figure 3. The 14 binary periodic tasks likewise yield no gold passes. In multiclass periodic tasks, 27B reaches gold on 8/130, 3/130, and 21/190 episodes at frequencies 4, 5, and 6, respectively; 35B-A3B reaches gold on 5/130, 2/130, and 7/190. Success is heteroge-

neous and nonmonotonic across settings: mechanism parameters alone do not determine difficulty when noise, sample size, and representation also vary. This motivates instance-level calibration and examining which families supply demonstrations.

5.5 WHICH TASKS SUPPLY TRAINING DEMONSTRATIONS?

Successful predictions are only the first selection step. The 6,000 requested episodes yield 3,881 valid scores and 916 gold passes. Of these, 468 gold-qualified episodes time out; 448 satisfy score and completion checks. Interaction and content screening leaves 422 automatic candidates, and final review retains 413 histories (6.9% of requests). At the common execution budget, 27B produces 2,361 valid scores and 35B-A3B produces 1,520; wall-time exits without valid predictions number 632 and 1,464, respectively. Demonstration yield therefore depends on finishing a usable interaction as well as reaching a high score. Appendix C details the screening stages.

Table 3: **Structural challenge and demonstration yield.** H is structural-reference benefit and $M = u_m(g) - u_m(c_s)$ is gold's margin above ordinary. Gold rates retain all requested episodes in the denominator. Candidates are counted after automatic screening and before final review.

Stratum	Tasks	Gold / requested	Candidates	Candidate tasks
$H \leq 0$	86	833 / 1,720	396	83
$H > 0, M \leq 0$	4	14 / 80	5	3
$M > 0$	210	69 / 4,200	21	15

Structural challenge limits demonstration coverage. Positive-H tasks constitute 71.3% of the collection but supply only 26/422 (6.2%) automatic candidates. Tasks with nonpositive H reach gold in 48.4% of requested episodes, compared with 1.6% for tasks whose gold exceeds the ordinary reference (Table 3). No parity trajectory survives the automatic screen. Demonstrations thus cover different structure from the generated corpus, identifying a collection priority: tasks whose reference gains are seldom realized by agents.

Task coverage and trajectory count measure different constraints. Automatic candidates cover 101/300 tasks (33.7%): 18/214 (8.4%) with positive H and 83/86 (96.5%) with nonpositive H. Repeated collection adds qualifying histories from a narrow task set. More histories supply further experimental decisions within covered tasks; broader mechanism coverage requires success on currently uncovered tasks. Both counts characterize the resulting supervision.

Revision within retained histories. Retained histories have medians of 29,006 tokens and 28 executor events, versus 35,202.5 tokens and 31 events across 5,992 parseable trajectories (Appendix F.5). They preserve substantial interaction at a lower median length. One agent parses date and string fields, fixes an estimator argument, narrows a timed-out search, and submits a two-model ensemble, obtaining test AUC 0.7744 above gold 0.7662 (Appendix E). The history supervises recovery and final selection alongside solution implementation.

5.6 CONTINUOUS HISTORY VERSUS INDEPENDENT RESTARTS

We compare one continuous 30-minute/60-turn run with two fresh 15-minute/30-turn runs on 12 held-out tasks. The independent submissions are selected by public-validation score before hidden grading. Both modes use feedback and the same cumulative time and turn budgets; their token costs can differ (Zhao et al., 2026).

Figure 5 reveals different effects across models. For 27B, continuous execution succeeds on 2/12 tasks before and after SFT; independent execution changes from 1/12 to 0/12. Its $+8.33$ pp change in continuity advantage therefore comes from lower independent success. For 35B-A3B, continuous execution remains at 0/12 while independent execution improves from 0/12 to 1/12, producing a -8.33 pp change. Both paired intervals include zero. External transfer gains therefore coexist with model-dependent responses to budget allocation; they do not establish a larger advantage for continuous history.

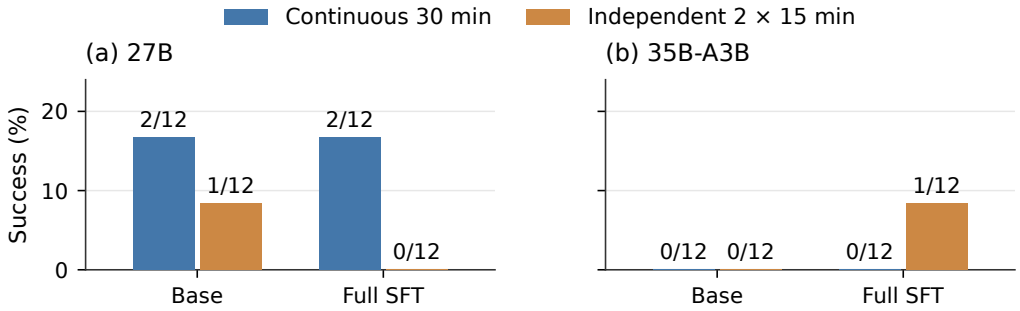


Figure 5: **Continuous versus independent experimentation.** Bars show gold success rates and successful tasks out of 12. Independent submissions are selected using public validation. Table 15 reports paired task-bootstrap intervals.

6 Related work

Synthetic environments and training data. Executable environments support agent post-training in general interaction, software engineering, and ML experimentation (Xi et al., 2025; Pan et al., 2025; Nathani et al., 2025; Qiang et al., 2025). Task generators extend this approach through difficulty control and verification (Hu et al., 2025; Zala et al., 2024; Shi et al., 2026; Fang et al., 2026; Zhang et al., 2026); procedural generation and unsupervised environment design provide related precedents (Cobbe et al., 2020; Dennis et al., 2020). Closest to IterLab, ML-AutoResearch selects teacher trajectories on automatically generated environments, Synthetic Sandbox constructs verifiable MLE tasks, and MLE-Smith automates generate–verify–execute construction (Cai et al., 2026; Zhou et al., 2026; Qiang et al., 2026). TRACE uses hidden interventions to derive diagnostic rewards, while SPADE adapts environments through designer–solver self-play (Sun et al., 2026; Liu et al., 2026). IterLab focuses on task-side measurements: paired ordinary and structure-enhanced references quantify structural benefit, and recovery checks assess performance retention after representation changes.

Learning from execution and feedback. Iterative revision and execution-guided search improve candidate solutions using feedback (Madaan et al., 2023; Shinn et al., 2023; Gou et al., 2024; Chen et al., 2024; Zhou et al., 2024; Romera-Paredes et al., 2024; Novikov et al., 2025; Agrawal et al., 2026; Toledo et al., 2025). Experience can also be consolidated into reusable memories or skills (Zhao et al., 2024; 2026), while agent tuning learns from selected parts of interaction histories (Zeng et al., 2024; Chen et al., 2023; 2025; Ma et al., 2026b). IterLab supplies complete histories of ML experiments and uses terminal quality and interaction integrity to select supervision. Its task diagnostics characterize which generated challenges actually contribute demonstrations.

Evaluating ML and software agents. MLAgentBench, DSBench, and MLE-bench evaluate prediction modeling (Huang et al., 2024; Jing et al., 2025; Chan et al., 2025); SWE-fficiency tests software optimization (Ma et al., 2026a). These benchmarks measure transfer; our matched-budget study compares sustained experimentation with fresh starts.

7 Discussion and conclusion

IterLab connects calibrated task construction with training on complete experimental histories. Paired references and recovery checks identify improvement opportunities; selected histories preserve how agents use feedback to pursue them. Training improves both models on four external benchmarks, including software optimization beyond the synthetic prediction setting. Structural tasks dominate generation but supply few demonstrations; stronger collectors could broaden the experimental decisions represented in training. The matched-budget study finds model-dependent effects of continuous history. Together, these analyses connect task design, supervision, and transfer when training agents to experiment and improve solutions over long horizons.

AI USE STATEMENT

We used OpenAI Codex and Kimi K3 in the preparation of this work, covering the following activities.

Writing assistance and polishing. OpenAI Codex helped refine academic English, improve paragraph organization, clarify terminology, and check consistency across the abstract, main text, tables, captions, and appendix.

Retrieval and discovery. OpenAI Codex assisted with finding related literature and official benchmark documentation, comparing published evaluation protocols, and organizing sources for the related-work discussion and experimental setup.

Research ideation and execution. OpenAI Codex supported the formulation of experimental comparisons and diagnostics, revision of analysis and visualization code, analysis of supplied experimental records, mathematical presentation, and checks of numerical and reporting consistency.

Drafting paper sections. OpenAI Codex assisted in drafting and revising sections of the manuscript from the authors' methods, results, and instructions, including the abstract, introduction, method description, experimental analysis, related work, and conclusion.

Synthetic dataset generation. Kimi K3 generated fictional application narratives for the synthetic tasks from predefined domain, scenario, angle, and audience fields. The narratives were combined with programmatically generated prediction data and deterministically rendered task facts. Data values, labels, generating mechanisms, and private calibration information were not provided to the narrative model. Appendix B describes this process and its prompts.

The authors are responsible for the scientific claims, reported results, citations, and final content of the manuscript.

REPRODUCIBILITY STATEMENT

Appendices A–F specify task generation, calibration, prompts, demonstration screening, SFT settings, and evaluation protocols. The source package includes aggregate tables and editable figure sources.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. Gepa: Reflective prompt evolution can outperform reinforcement learning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 8479–8565, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/0e9e708b6f48e14fd0ac29e167413f76-Paper-Conference.pdf.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pp. 2623–2631, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330701. URL https://doi.org/10.1145/3292500.3330701.

Ziyang Cai, Amir Saeidi, and Harkirat Behl. Ml-autoresearch: Training machine learning research agents with automatically generated environments, 2026. URL https://arxiv.org/abs/2603.17216.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Aleksander Madry,

and Lilian Weng. Mle-bench: Evaluating machine learning agents on machine learning engineering. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 50466–50494, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/7e3767db483c942b883eb4f8cfb74e31-Paper-Conference.pdf.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fireact: Toward language agent fine-tuning, 2023. URL https://arxiv.org/abs/2310.05915.

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, pp. 785–794, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939785. URL https://doi.org/10.1145/2939672.2939785.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost, 2016. URL https://arxiv.org/abs/1604.06174.

Xinyun Chen, Maxwell Lin, Nathanael Schaerli, and Denny Zhou. Teaching large language models to self-debug. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 8746–8825, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/2460396f2d0d421885997dd1612ac56b-Paper-Conference.pdf.

Zhixun Chen, Ming Li, Yuxuan Huang, Yali Du, Meng Fang, and Tianyi Zhou. ATLAS: Agent tuning via learning critical steps. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 25334–25349, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1299. URL https://aclanthology.org/2025.findings-acl.1299/.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2048–2056. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/cobbe20a.html.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 35549–35562, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/98ed250b203d1ac6b24bbcf263e3d4a7-Paper-Conference.pdf.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 13049–13061. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/985e9a46e10005356bbaf194249f6856-Paper.pdf.

B. Efron. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1): 1 – 26, 1979. doi: 10.1214/aos/1176344552. URL https://doi.org/10.1214/aos/1176344552.

Runnan Fang, Shihao Cai, Baixuan Li, Jialong Wu, Guangyu Li, Wenbiao Yin, Xinyu Wang, Xiaobin Wang, Liangcai Su, Zhen Zhang, Shibin Wu, Zhengwei Tao, Yong Jiang, Pengjun Xie, Ningyu Zhang, Fei Huang, Wentao Zhang, and Jingren Zhou. Towards general agentic intelligence via environment scaling. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17610–17621, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.872. URL https://aclanthology.org/2026.findings-acl.872/.

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189 – 1232, 2001. doi: 10.1214/aos/1013203451. URL https://doi.org/10.1214/aos/1013203451.

Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-interactive critiquing. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 57734–57811, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/fef126561bbf9d4467dbb8d27334b8fe-Paper-Conference.pdf.

James A. Hanley and Barbara J. McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982. doi: 10.1148/radiology.143.1.7063747.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, et al. Array programming with NumPy. *Nature*, 585:357–362, 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637:319–326, 2025. doi: 10.1038/s41586-024-08328-6. URL https://doi.org/10.1038/s41586-024-08328-6.

Mengkang Hu, Pu Zhao, Can Xu, Qingfeng Sun, Jian-Guang Lou, Qingwei Lin, Ping Luo, and Saravan Rajmohan. Agentgen: Enhancing planning abilities for large language model based agent via environment and task generation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, pp. 496–507, New York, NY, USA, 2025. Association for Computing Machinery. doi: 10.1145/3690624.3709321. URL https://doi.org/10.1145/3690624.3709321.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating language agents on machine learning experimentation. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 20271–20309. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/huang24y.html.

Liqiang Jing, Zhehui Huang, Xiaoyang Wang, Wenlin Yao, Wenhao Yu, Kaixin Ma, Hongming Zhang, Xinya Du, and Dong Yu. Dsbench: How far are data science agents from becoming data science experts? In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 32597–32649, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/50e9ad960ae78b741a6b4fea533f2eaf-Paper-Conference.pdf.

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. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

Kimi Team et al. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Vijay Anand Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing activation recomputation in large transformer models. In D. Song, M. Carbin, and T. Chen

(eds.), *Proceedings of Machine Learning and Systems*, volume 5, pp. 341–353. Curran, 2023. URL https://proceedings.mlsys.org/paper_files/paper/2023/file/80083951326cf5b35e5100260d64ed81-Paper-mlsys2023.pdf.

Bo Liu, Simon Yu, Yiding Jiang, Ao Qu, Andrew Zhao, Zichen Liu, Junsu Kim, Zijian Zhou, Seungone Kim, Tongzheng Ren, Mickel Liu, Hanfei Yu, Zhaorun Chen, Weiyan Shi, Paul Pu Liang, Luke Zettlemoyer, Yejin Choi, and Natasha Jaques. Spade: Self-play in adaptive synthetic executable environments, 2026. URL https://arxiv.org/abs/2608.19197.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017. URL https://arxiv.org/abs/1608.03983.

Jeffrey Jian Ma, Milad Hashemi, Amir Yazdanbakhsh, Kevin Swersky, Ofir Press, Enhui Li, Vijay Janapa Reddi, and Parthasarathy Ranganathan. Swe-fficiency: Can language models optimize real-world repositories on real workloads?, 2026a. URL https://arxiv.org/abs/2511.06090.

Murong Ma, Tianyu Chen, Yun Lin, Shuai Lu, Qinglin Zhu, Yeyun Gong, Zhiyong Huang, Peng Cheng, Yan Lu, and Jin Song Dong. From patches to trajectories: Privileged process supervision for software-engineering agents, 2026b. URL https://arxiv.org/abs/2605.21996.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594. Curran Associates, Inc., 2023. doi: 10.52202/075280-2019. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91edff07232fb1b55a505a9e9f6c0ff3-Paper-Conference.pdf.

M. D. McKay, R. J. Beckman, and W. J. Conover. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979. doi: 10.1080/00401706.1979.10489755. URL https://doi.org/10.1080/00401706.1979.10489755.

Wes McKinney. Data structures for statistical computing in python. *SciPy 2010*, 2010. doi: 10.25080/Majora-92bf1922-00a. URL https://doi.org/10.25080/Majora-92bf1922-00a.

Deepak Nathani, Lovish Madaan, Nicholas Roberts, Nikolay Bashlykov, Ajay Menon, Vincent Moens, Mikhail Plekhanov, Amar Budhiraja, Despoina Magka, Vladislav Vorotilov, Gaurav Chaurasia, Dieuwke Hupkes, Ricardo Silveira Cabral, Tatiana Shavrina, Jakob Nicolaus Foerster, Yoram Bachrach, William Yang Wang, and Roberta Raileanu. MLGym: A new framework and benchmark for advancing AI research agents. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=ryTr83DxRq.

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

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with SWE-gym. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 47717–47737. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/pan25g.html.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas,

Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85): 2825–2830, 2011. URL http://jmlr.org/papers/v12/pedregosa11a.html.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/14491b756b3a51daac41c24863285549-Paper.pdf.

Rushi Qiang, Yuchen Zhuang, Yinghao Li, Dingu Sagar V K, Rongzhi Zhang, Changhao Li, Ian Shu-Hei Wong, Sherry Yang, Percy Liang, Chao Zhang, and Bo Dai. Mle-dojo: Interactive environments for empowering llm agents in machine learning engineering, 2025. URL https://arxiv.org/abs/2505.07782.

Rushi Qiang, Yuchen Zhuang, Anikait Singh, Percy Liang, Chao Zhang, Sherry Yang, and Bo Dai. Mle-smith: Scaling mle tasks with automated multi-agent pipeline. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 5963–5990, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/0abcc0198bef043543d8837cfc1e18f7-Paper-Conference.pdf.

Qwen Team. Qwen3.6-27b. Hugging Face model repository, 2026a. URL https://huggingface.co/Qwen/Qwen3.6-27B.

Qwen Team. Qwen3.6-35b-a3b. Hugging Face model repository, 2026b. URL https://huggingface.co/Qwen/Qwen3.6-35B-A3B.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–16, 2020. doi: 10.1109/SC41405.2020.00024.

Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. ZeRO-Offload: Democratizing Billion-Scale model training. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pp. 551–564. USENIX Association, July 2021. ISBN 978-1-939133-23-6. URL https://www.usenix.org/conference/atc21/presentation/ren-jie.

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

Dingfeng Shi, Jingyi Cao, Qianben Chen, Weichen Sun, Weizhen Li, Hongxuan Lu, Fangchen Dong, Tianrui Qin, Zhu, Minghao Liu, Yuchen Jiang, Jian Yang, Ge Zhang, JIAHENG LIU, Changwang Zhang, Jun Wang, and Wangchunshu Zhou. Taskcraft: Automated generation of agentic tasks. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 43714–43734, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/48644509339cb3076f7b0407c7588af6-Paper-Conference.pdf.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.

Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2009.03.002. URL https://www.sciencedirect.com/science/article/pii/S0306457309000259.

Rui Sun, Zhan Shi, and Bing He. Trace: Training reasoning agents for causal exploration with synthesized rewards, 2026. URL https://arxiv.org/abs/2609.10315.

Edan Toledo, Karen Hambardzumyan, Martin Josifoski, Rishi Hazra, Nicolas Baldwin, Alexis Audran-Reiss, Michael Kuchnik, Despoina Magka, Minqi Jiang, Alisia Lupidi, Andrei Lupu, Roberta Raileanu, Tatiana Shavrina, Kelvin Niu, Jean-Christophe Gagnon-Audet, Michael Shvartsman, Shagun Sodhani, Alexander Miller, Abhishek Charnalia, Derek Dunfield, Carole-Jean Wu, Pontus Lars Erik Saito Stenetorp, Nicola Cancedda, Jakob Foerster, and Yoram Bachrach. Ai research agents for machine learning: Search, exploration, and generalization in mle-bench. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 35309–35348. Curran Associates, Inc., 2025. doi: 10.52202/085713-1186. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/328b81881da145412f2bc56c998dfb6a-Paper-Conference.pdf.

David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(05)80023-1. URL https://www.sciencedirect.com/science/article/pii/S0893608005800231.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Xin Guo, Dingwen Yang, Chenyang Liao, Wei He, Songyang Gao, Lu Chen, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. AgentGym: Evaluating and training large language model-based agents across diverse environments. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27914–27961, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1355. URL https://aclanthology.org/2025.acl-long.1355/.

Abhay Zala, Jaemin Cho, Han Lin, Jaehong Yoon, and Mohit Bansal. Envgen: Generating and adapting environments via LLMs for training embodied agents. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=F9tqgOPXH5.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. AgentTuning: Enabling generalized agent abilities for LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3053–3077, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.181. URL https://aclanthology.org/2024.findings-acl.181/.

Jiayi Zhang, Fanqi Kong, Guibin Zhang, Maojia Song, Zhaoyang Yu, Jianhao Ruan, Jinyu Xiang, Bang Liu, Chenglin Wu, and Yuyu Luo. Scalable environments drive generalizable agents, 2026. URL https://arxiv.org/abs/2605.18181.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Thirty-Eighth AAAI Conference on Artificial Intelligence*, pp. 19632–19642. AAAI Press, 2024. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

Xinyu Zhao, Junpeng Wang, Yuzhong Chen, Menghai Pan, Chin-Chia Michael Yeh, Jiarui Sun, Yan Zheng, Mahashweta Das, and Tianlong Chen. Demystify the role of memory in machine learning engineering agents. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 10811–10826, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.525. URL https://aclanthology.org/2026.findings-acl.525/.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62138–62160. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/zhou24r.html.

Yuhang Zhou, Lizhu Zhang, Yifan Wu, Jiayi Liu, Xiangjun Fan, Zhuokai Zhao, and Hong Yan. Synthetic sandbox for training machine learning engineering agents, 2026. URL https://arxiv.org/abs/2604.04872.

A Task construction and reference diagnostics

The main collection contains 300 independently generated prediction tasks. Each combines a calibrated DGP with representation transformations, row permutation, identifier remapping, and a contextual description. Appendix D describes how the collected and supplemental demonstrations form the training set.

Construction diagnostics use the admitted task configurations and fitted references. Rollout accounting follows each stage from requested episodes to automatic candidates and reviewed training demonstrations.

Task supply. The collection spans six mechanism families and 100 tasks per type (Table 4). Appendix A.1 records family counts and generation outcomes. Narrative generation leaves the task data and calibration files unchanged.

Table 4: **Composition and reference calibration.** Each row represents 100 independent tasks. H measures structural-reference benefit; $M > 0$ means gold requires better performance than the strong mean. Differences retain metric-specific units.

Task / metric	Tasks	Training rows	Median H	$M > 0$
Binary / AUC	100	809–4,519	0.1665	72 / 100
4-class / macro-F1	100	868–4,528	0.0282	66 / 100
Regression / RMSE	100	1,030–4,496	0.2703	72 / 100

Feasibility checks and improvement opportunities. The structure-enhanced reference outperforms strong on 214 tasks; 210 require better performance than strong to reach gold. All tasks pass the recovery check, with zero median recovery loss within each metric. For the main-text figure, each utility difference is divided by $d_t = u_m(c_o) - u_m(c_w) > 0.05$. Thus the plotted gaps are H/d_t, $[u_m(g) - u_m(c_s)]/d_t$, $[u_m(c_n) - u_m(c_s)]/d_t$, and $[u_m(c_r) - u_m(c_s)]/d_t$. Values are neither clipped nor pooled in incompatible raw metric units. Figure 6 separates structural and representation effects; Appendix A.3 gives their descriptive summaries.

Realized challenge. Appendix F.4 reports controlled difficulty sweeps and task-level profiles by mechanism setting. Agent success rates measure realized challenge under the specified execution budget.

Available supervision. The 6,000 requests yield 3,881 valid scores, 916 gold passes, and 422 automatic candidates from 101 tasks (Appendix C). Only 26 candidates come from positive-H tasks, including 21 from positive-M tasks. This concentration concerns task properties, not the presence of useful revisions within a history, and distinguishes generated tasks from available supervision.

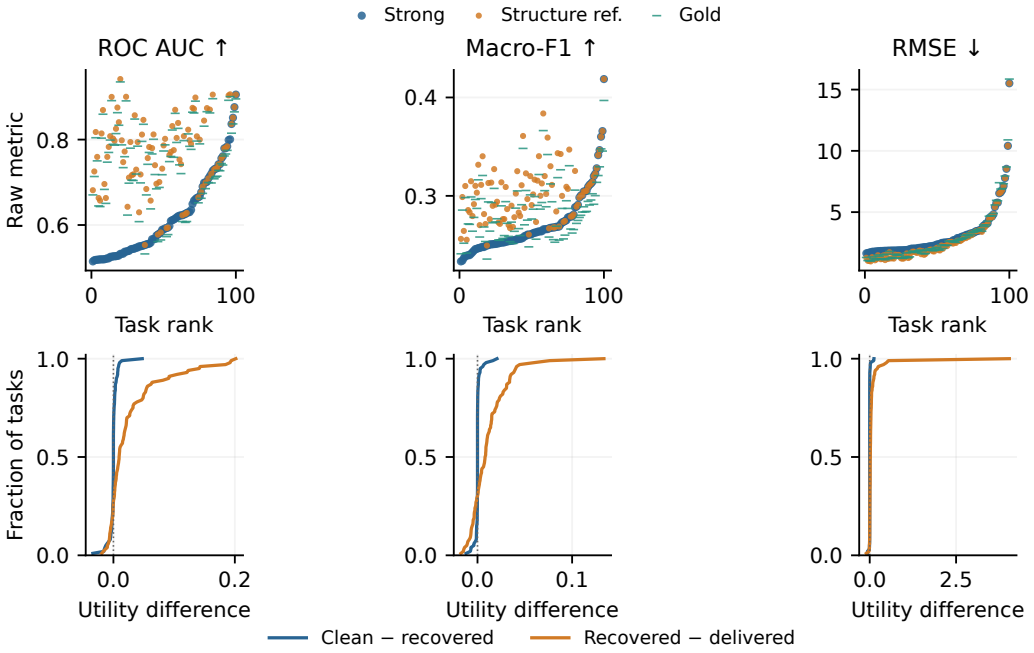


Figure 6: **Structural assistance and representation effects.** Top: strong, structure-enhanced, and gold scores, with 100 tasks per metric ordered by strong score. Bottom: task-level distributions of recovery loss L and representation benefit V in utility units. These comparisons assess structural benefits and whether processing restores reference performance; agent execution measures realized challenge.

Table 5: **Construction choices, agent decisions, and checks.** Mechanisms and statistical conditions define the prediction problem; feature engineering changes input-processing demands; task descriptions organize the public requirements.

Construction choice	Decision facing the agent	Check on the task
Function family, frequency, interaction order	Which features and models capture the relationship?	Ordinary versus structure-enhanced scores
Samples, noise, redundancy, class balance	How to fit models and compare validation results?	Separation from weak and score dispersion
Numerical and cross-column encodings, formats, irrelevant fields	How to parse, combine, and select useful inputs?	Reference performance before and after recovery
Application background, data and evaluation requirements	How to formulate validation and experiments?	Task facts rendered from the public contract

A.1 DGP FAMILIES AND DIFFICULTY PARAMETERS

Algorithm 1: Controllable task construction

1. Choose a seeded mechanism, requested statistical parameters, and representation intensity.
2. Screen statistical changes with the proxy, then check the full portfolio. Traverse the finite repair queue; reject if the final guardrails fail.
3. Estimate weak, strong, and structure-enhanced references; set gold by Eq. (1) and record Bayes diagnostics.
4. Apply Stage B; use recorded inverses to recover features and check Eq. (7).
5. Measure the delivered-data reference; verify the weak-to-structure span and calibration verdict.
6. Shuffle rows, remap identifiers, and check public/private file alignment.
7. Add contextual prose and deterministic public facts; publish the admitted task and committed parameters.

The generator uses purely synthetic features and targets. A deterministic seed selects a family from a type-dependent weighted pool, with more parity tasks for binary classification and more periodic tasks for multiclass classification and regression. Polynomial, saturating, and threshold families provide controls. Base datasets contain 3,000–7,000 rows before train subsampling, with a test fraction of 0.35. Most specifications begin with six features, four informative; parity specifications begin with eight features, six informative. Redundant columns and Stage B change the delivered feature count. Multiclass tasks use four classes. The collection contains 112 hidden-periodic, 60 rotated-radial, 42 parity, 38 polynomial, 29 threshold, and 19 saturating tasks.

Table 6: Difficulty design ranges before repair. C denotes classification and R regression. Family-specific caps and repair can reduce the requested values.

Parameter	Design range	Applicability
Label noise	$[0, 0.20]$	C, R
Boundary concentration	$[0, 2]$	C
Class separation	$[0.3, 2]$	C
Nonlinearity level	Integers 0–4	C, R
Redundant features	Integers 0–12	C, R
Class imbalance	$[1, 10]$	C
Feature-noise scale	$[0, 1.2]$	C, R
Feature-noise heterogeneity	$[0, 2]$	C, R
Training subsample	800–3, 000, or no subsampling	C, R
Regression noise multiplier	$[1, 5]$	R
Heteroscedasticity	$[0, 2.5]$	R
Heavy-tail parameter	$[0, 3]$	R
Target-outlier fraction	$[0, 0.25]$	R

The sampler supports Latin hypercube sampling (McKay et al., 1979), but the current builder requests one vector per seed. This produces independent seeded vectors, without demonstrated corpus-wide Latin hypercube stratification. Periodic frequency varies over $\{4, 5, 6\}$ and parity order over $\{4, 5\}$. Feature noise is capped at 0.15, with its heterogeneity set to zero, for parity, hidden-periodic, and rotated-radial families. These caps retain signals fragile to perturbations of the informative coordinates.

The proxy order is label noise, boundary concentration, class separation, nonlinearity, redundant features, imbalance, feature noise, feature-noise heterogeneity, regression noise, heteroscedasticity, heavy tails, target outliers, lossy missingness, and training subsampling. Irrelevant parameters are skipped. The full check applies a directional reference-to-weak separation guardrail relative to estimated dispersion and a 0.98 upper guardrail for bounded classification metrics. A finite fallback queue can undo committed changes. The bounded search terminates, with rejection available as an outcome.

More precisely, the full Stage A guardrail requires

$$u_m(c_s) - u_m(c_w) > k\sigma_s, \quad k = 1, \qquad c_s \leq 0.98 \quad \text{for bounded classification metrics.} \tag{3}$$

The proxy uses its own cross-validation estimate and uncertainty in the same directional comparison. Typical decreasing-noise ladders are $(t, 0.66t, 0.33t, 0)$; nonlinearity and redundancy descend in integer steps to zero. Imbalance and regression noise use $(t, 1 + (t-1)/2, 1)$. Subsampling uses $(n, \lfloor 1.5n \rfloor, 2n, \text{all})$. Class separation uses the sorted unique set of $(t, 1.5t, 2t, t_0)$ when $t < t_0$, otherwise just the default t_0. The heavy fallback precomputes these ladders from the committed configuration, traverses active parameters in reverse proxy order, skips each current rung, and appends the default if needed. Precomputing prevents repeated fractional relaxation from approaching a default indefinitely.

The history contains 332 outcome records: 300 accepted, 25 rejected, and seven infrastructure errors. Of the rejections, 24 fail the weak-to-structure span gate and one fails the full separation guardrail. The 300 accepted slots used candidate index zero in 273 cases, index one in 22, and index two in five. These indices describe the predeclared candidate sequence per slot; infrastructure retries remain distinct from scientific rejections.

A.2 Reference estimation and statistical definitions

For the synthetic metrics, $u_m(s) = d_m s$, where $d_m = 1$ for ROC AUC and macro-F1 and $d_m = -1$ for RMSE (Hanley & McNeil, 1982; Sokolova & Lapalme, 2009). Larger utility is better, with differences retaining metric-specific units. With weak, ordinary strong, structure-enhanced, and gold scores denoted by c_w, c_s, c_o, g, respectively,

$$H = u_m(c_o) - u_m(c_s), \qquad M = u_m(g) - u_m(c_s). \tag{4}$$

H is the structural-reference benefit and M is gold's utility difference from strong. They can have different signs after the dispersion adjustment in Eq. (1).

The weak classifier predicts the training majority class; a constant score has AUC 0.5 (Hanley & McNeil, 1982). The weak regressor predicts the training mean. The strong portfolio includes linear and polynomial models, tree ensembles, histogram gradient boosting, nearest-neighbor, kernel, and neural-network estimators (Pedregosa et al., 2011; Friedman, 2001), with XGBoost (Chen & Guestrin, 2016) and LightGBM (Ke et al., 2017) when installed. Generation receipts record NumPy 1.26.4, pandas 2.2.3, scikit-learn 1.5.2, XGBoost-CPU 2.1.4, and LightGBM 4.6.0 (Harris et al., 2020; McKinney, 2010; Pedregosa et al., 2011; Chen & Guestrin, 2016; Ke et al., 2017). CatBoost, TabPFN, and Optuna were unavailable (Prokhorenkova et al., 2018; Hollmann et al., 2025; Akiba et al., 2019).

The heavy search uses eight coarse candidates per family, refines the top two with a fine budget of 16 candidates and five-fold cross-validation, and constructs a stacking estimator from leading families (Wolpert, 1992). Classification training bootstraps sample within class; regression training and test-row bootstraps sample with replacement (Efron, 1979). For bootstrap b, define

$$z_b = \max\{u_m(s_b^{\text{single}}), u_m(s_b^{\text{ensemble}})\}, \quad \bar{z} = \frac{1}{B'}\sum_{b\in\mathcal{B}} z_b, \quad \sigma = \sqrt{\frac{1}{B'-1}\sum_{b\in\mathcal{B}}(z_b - \bar{z})^2}, \tag{5}$$

where $\mathcal{B}$ contains successful finite replicates and $B' = |\mathcal{B}| \leq 40$. If one estimator fails, the other finite score can be used; if both fail, the replicate is omitted. A completely failed estimate is nonfinite. Dispersion is zero with only one surviving replicate. The final reference statistics are computed from these bootstrap replicates.

Privileged features come from the DGP fingerprint: rotated coordinates, parity interactions with soft sign treatment under measurement noise, and periodic or radial transformations at the known frequency. Control families may add no privileged columns. The code does not enforce reference ordering; the corpus includes ties and reversals. The separate Bayes diagnostic uses 60,000 Monte Carlo samples in this run and includes approximations under observation noise.

The stored field `se` is the bootstrap sample standard deviation, not the standard error. Gold is rounded to six decimals after Eq. (1). Admission requires a finite weak-to-structure utility span above 0.05 and a calibration verdict other than `unfair`. On admitted tasks the anchor exceeds

weak, so $|a - w| = a - w$. Neither positive H nor positive M is required for every task. Bayes diagnostics do not determine gold or trigger repair.

For a hidden-periodic task, the numeric latent component and added features are

$$f_*(z) = 2\sin(\omega v^\top z) + 0.3v^\top z, \qquad \phi(x) = [\sin(\omega v^\top x), \cos(\omega v^\top x), v^\top x]. \tag{6}$$

Here z is the clean informative vector and x its possibly noisy observation. Label generation adds categorical effects and applies task-specific labeling and noise. The generator knows the direction v and frequency ω. These features expose a useful modeling route to the fitted reference; they are not supplied to the agent.

The numeric latent components of two other structural families make the comparison explicit. With an orthogonal matrix R fixed by the DGP, parity uses $f_*(z) = 3\prod_{j\in I}\operatorname{sign}((Rz)_j)$ for four or five selected coordinates. The augmented portfolio receives rotated observations and a product of soft signed posterior scores under measurement noise. For rotated-radial tasks, $r(z) = \|(Rz)_{1:\min(3,k)}\|_2$ and $f_*(z) = 2\sin(0.8\omega r(z)) + 0.2w^\top z$; augmentation supplies $r(x)$ and $\sin(0.8\omega r(x))$. The hidden rotation, direction, and frequency are generator-side information. They illustrate what the reference measures, rather than a procedure already recovered by the public agent.

Lower medal tiers interpolate between gold and weak: $s_{\text{silver}} = g + (c_w - g)/3$, $s_{\text{bronze}} = g + 2(c_w - g)/3$, and $s_{\text{median}} = c_w$. These are private grading metadata. Agents receive the metric and direction without numeric thresholds.

A.3 Stage B operators

Table 7: Observed Stage B operators. Counts are operator occurrences, not mutually exclusive task counts. Recovery uses recorded parameters.

Operator	Count	Transformation and recovery information
Opaque rename	295	Replace feature names; retain the mapping.
Categorical recoding	169	Apply a shared category-code map across splits.
Noise distractor (`leak_trap`)	167	Add independent noise; inverse removes the column.
Merge columns	147	Combine values in a string; inverse parses them.
Recoverable missingness	94	Encode missing values with reconstruction information.
Power encoding	90	Apply a sampled map and its recorded inverse.
Split ratio	85	Replace a value by numerator and denominator columns.
Linear mixing	84	Mix a pair with a stored invertible matrix.
Date encoding	81	Encode a numeric coordinate as a date.
Unit scaling	75	Multiply by a stored conversion factor.
Affine shift	74	Apply stored scale and offset.
Exponential/log encoding	73	Deliver an exponential; recover with a logarithm.
Bucket with residual	70	Retain bucket identity and residual information.
Conditional encoding	38	Apply a conditional map with recorded parameters.

Let c_n be strong measured directly on delivered inputs and c_r after recorded recovery. With their estimated dispersions, admission requires

$$u_m(c_r) \geq u_m(c_s) - 2\sqrt{\sigma_s^2 + \sigma_r^2}. \tag{7}$$

Recovery loss is $L = u_m(c_s) - u_m(c_r)$ and representation benefit is $V = u_m(c_r) - u_m(c_n)$. The one-sided check allows estimator variation and rounding; it checks reference performance retention, not exact value-by-value inversion or public discoverability.

Stage B sampling intensity is 0.5. Some operators round values, so exact round-trip equality is not general. Every published task passes Eq. (7). Observed recovery-loss ranges are $[-0.033861, 0.047491]$ AUC, $[-0.012062, 0.020611]$ macro-F1, and $[-0.067902, 0.122584]$ RMSE utility units. Negative loss indicates a recovered reference score above the clean score. These are descriptive extrema, not uncertainty intervals. Median recovery loss is zero within each metric. Median representation benefits are 0.0095 AUC, 0.0083 macro-F1, and 0.0172 RMSE utility units; 73, 70, and 89 tasks, respectively, have positive benefits.

A.4 A WORKED CALIBRATION EXAMPLE

Task `k3d_task0021_57088d5eff9a` is a binary hidden-periodic instance with frequency 5, 3,924 training rows, 2,113 test rows, and 17 delivered features. Table 8 reports its frozen reference measurements. Stage B applies a bucket-with-residual encoding, power encoding, column merging, categorical recoding, an independent-noise distractor, and opaque renaming.

Table 8: **Task0021: separating processing from further modeling.** Scores are AUC; differences are AUC utility units. Values are shown to six decimals. Each score is measured with the fitted reference at the indicated representation or feature setting.

Measurement	Value	Interpretation
Weak c_w	0.500000	Constant baseline
Strong c_s	0.699110	Portfolio on ordinary Stage A features
Structure-enhanced c_o	0.844410	Portfolio with DGP-derived periodic features
Gold g	0.831357	Target calibrated from the structural anchor
Structural benefit H	0.145300	Structure-enhanced minus strong
Gold margin M	0.132247	Gold minus strong
Delivered-data reference c_n	0.505288	Direct fitting after Stage B
Recovered strong c_r	0.699102	Fitting after recorded inverse operations
Recovery loss L	0.000008	Strong minus recovered
Representation benefit V	0.193814	Recovered minus delivered-data reference

Recorded recovery restores the ordinary reference to within 0.000008 AUC of its original score. Gold lies a further 0.132255 AUC above the recovered score, separating the input-processing demand from the remaining modeling demand. The structure-enhanced reference quantifies a beneficial feature representation available from the known mechanism; the agent must search for a successful solution using public inputs.

B PROMPTS AND PUBLIC INFORMATION

B.1 NARRATIVE GENERATION

Kimi K3 (Kimi Team et al., 2026) writes fictional context from predefined domain, scenario, angle, and audience fields. It receives no data values, mechanism parameters, or private scores. A deterministic renderer appends public task facts. The 20×15 contextual design gives 300 narratives; scientific data and calibration are preserved. Narrative diversity and mechanism diversity are therefore reported separately.

The generator is `kimi-k3-test-dy`, with temperature 0.9, maximum output 8,192 tokens, and a 180-second request timeout. Transport retries use the same model. Three successive narrative-quality failures block a task, with no model or template fallback. The diversity check compares five-word-gram overlap with prior accepted texts at a maximum Jaccard threshold of 0.35 and checks title reuse.

The full system prompt is:

> Write only fictional business background in third person. You do not write competition instructions, discuss data or advise a reader. Return one JSON object only.

The user prompt substitutes only the four contextual fields into `{brief}`; a retry may append previous content-validation feedback.

> Write an English fictional business vignette using this narrative brief: `{brief}`
>
> Return exactly four string fields: title, overview, workflow, motivation. Use a distinctive 4-14 word title; overview should be 70-130 words, workflow 35-80 words, motivation 30-70 words. Workflow means a scene of ordinary business activity, NOT a participant workflow or instructions. Make this specific to the domain and focus; mention both domain and focus verbatim somewhere in the prose. Follow the writing angle, varying sentence structure and opening. Do not reuse generic organization or historical-dataset boilerplate. Use conditional language for business benefits.

> Use only third person business narration; never address a reader, give commands, tell someone what to do, or describe competition procedures. Do not mention any competition, challenge, benchmark, participant, submission, leaderboard, scoring, CSV, columns, target, data, dataset, records, metric, prediction, model, algorithm, learning, machine learning, statistical technique, calibration or evaluation. Do not discuss an objective to be solved by the reader. The surrounding application will supply all factual instructions separately. Do not include numbers, formulas, code or Markdown headings. Return only the four requested prose fields.

The renderer declares the narrative fictional and the features anonymized, then adds measured public row counts, schema, objective, and submission format. Narrative diversity is separate from DGP diversity.

B.2 Agent task prompt

This is the complete task prompt for the binary example in Appendix E. The scaffold adds its tool interface and system instructions, which must also be preserved in an anonymous release.

> You are given a tabular machine-learning competition task.
>
> Files in your working directory:
> `description.md`: the task brief — READ THIS FIRST
> `train.csv`: 3624 labelled rows (feature columns + `target`)
> `test.csv`: 1952 unlabelled rows (same features, no `target`)
> `sample_submission.csv`: the exact submission format
>
> Your job: train a model on `train.csv` and predict `target` for every row of `test.csv`.
>
> DELIVERABLE — this is what gets graded: Write your predictions to `submission.csv` in this directory. It must have exactly two columns, `id` and `target`, and one row for every id in `test.csv` (1952 rows plus the header).
>
> Scoring: roc_auc (higher is better). For this metric submit a continuous score/probability, not a hard 0/1 label.
>
> You may write and run any Python you like (pandas / numpy / scikit-learn are installed). The raw fields are delivered as-is: some are directly usable, some need transformation or decoding, and some are useless. Getting the representation right is the task.
>
> Write `submission.csv` before you finish.

Whitespace and list formatting are normalized; the source prompt is retained in the evidence package. Counts and metric instructions vary by task. Numeric gold, privileged features, and test labels are absent.

C Rollout accounting and trajectory screening

Collection uses temperature 0.7, top-p 0.95, and disabled thinking for both generating models. The 60-turn and 1,800-second limits apply to each episode. Infrastructure recovery is bounded; valid low scores and budget exhaustion do not trigger resampling. A submitted file can receive a valid score even when its episode subsequently times out. The tables report automatic screening; Appendix D describes the reviewed training export.

Normalized trajectory quality uses the weak-to-structure span:

$$q(s) = \frac{u_m(s) - u_m(c_w)}{u_m(c_o) - u_m(c_w)}, \qquad u_m(c_o) - u_m(c_w) > 0.05. \tag{8}$$

The audit requires $q(s) \geq 0.2$ in addition to strict gold, comparability, and completion checks. Unlike H, this denominator compares structure-enhanced to weak. A fitted reference can be exceeded, yielding $q > 1$.

Prediction regrading checks the submitted ID set, finite predictions, and agreement with the recorded score under the task's metric. Conversation checks require paired tool calls and results, at least two tool results, a complete final assistant response, and no episode timeout or forced-kill exit. Content screening holds suspicious private-grader access and row-order use for review. Exact normalized-content deduplication removes incidental tool IDs and workspace paths from the comparison; it is

not semantic deduplication. Intermediate command failures can remain when the overall episode completes successfully.

Table 9: **Rollout and automatic selection.** Score+budget includes strict gold, normalized quality, comparability, and completion checks. Candidates additionally pass conversation checks, content screening, and exact deduplication. Four 27B requests and one 35B request were blocked at the screening cutoff.

Model	Requested	Terminal	Scored	Gold	Score+budget	Candidates
27B	3,000	2,996	2,361	600	326	309
35B-A3B	3,000	2,999	1,520	316	122	113

Table 10: Per-type counts. Each model/type cell has 1,000 requested trajectories across 100 tasks. Scored and gold counts can include submissions left by an episode that subsequently timed out.

Model	Metric	Terminal	Scored	Gold	Candidates
27B	AUC	1,000	918	187	140
27B	Macro-F1	996	655	232	76
27B	RMSE	1,000	788	181	93
35B-A3B	AUC	1,000	561	129	55
35B-A3B	Macro-F1	999	380	88	21
35B-A3B	RMSE	1,000	579	99	37

Among 2,996 terminal 27B requests, outcome categories are 2,361 scored submissions, 632 wall-time budget exits without a valid score, two context-or-turn exits, and one other missing-valid-submission outcome. Among 2,999 terminal 35B requests, they are 1,520 scored submissions, 1,464 wall-time exits without a valid score, 13 context-or-turn exits, and two turn-budget exits. Scored submissions take precedence, so an episode can have a score and a timeout flag. The 468 gold-qualified timeout episodes explain the need to separate valid-score counts from SFT eligibility.

Of 448 score-and-budget-qualified trajectories, eight lack a complete final assistant response and 19 have row-order review flags. One belongs to both groups, leaving 422 candidates. Exact normalized-content deduplication removes none of these. Flagged histories are held for content review. The models' candidate sets cover 96 and 64 tasks, with 101 in their union. Keyword flags identify cases for review; the current execution setup provides logical separation of public inputs and private grading files. Final export includes manual content review and tokenization checks.

The candidate family counts are 173 polynomial, 115 threshold, 108 saturating, 16 hidden-periodic, ten rotated-radial, and zero parity. Only 6.2% of automatic candidates come from positive-H tasks, indicating limited demonstration coverage of the structural challenges.

Table 3 in the main text summarizes task coverage and automatic selection by structural benefit and gold margin.

Each request permits at most three infrastructure attempts. Repeat indices one to ten identify the planned model samples. Five requests were blocked at the screening cutoff and contribute neither valid scores nor zero-valued model outcomes.

D Training data and recipes

For a tokenized trajectory $x_{1:L}$, let $m_j = 1$ for assistant text and tool actions and $m_j = 0$ for all other tokens. Training minimizes

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = -\frac{\sum_{\tau \in \mathcal{D}_{\mathrm{train}}} \sum_{j=1}^{L_\tau} m_j \log \pi_\theta(x_j \mid x_{<j})}{\sum_{\tau \in \mathcal{D}_{\mathrm{train}}} \sum_{j=1}^{L_\tau} m_j}. \tag{9}$$

The converter preserves native tool-call pairing and order. Assistant text and actions receive supervision, while prompts and tool results remain in context with their loss masked (Zhou et al., 2026). The combined export contains 682 histories: one legacy probe, 258 from an earlier corpus, ten from

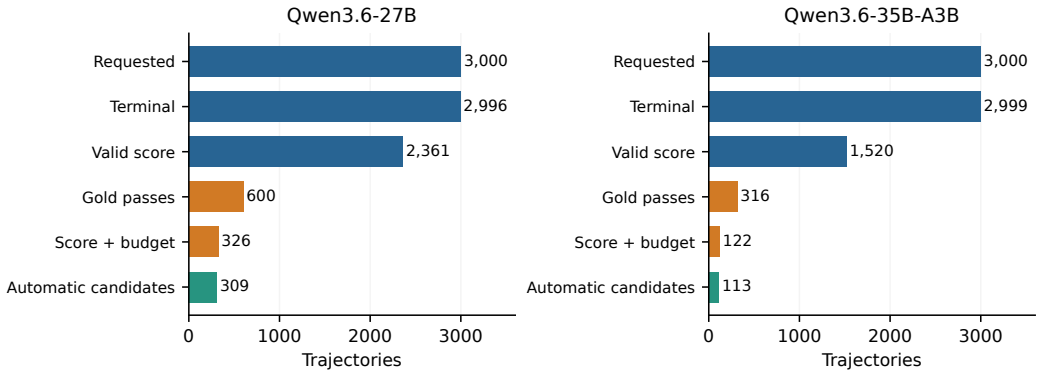


Figure 7: **Selection yield.** Counts follow the automatic screening steps in Section 4, before final demonstration review. Blocked requests remain in the requested-episode denominator.

the template-described collection, and 413 from the main collection. Its seed-42 random trajectory split assigns 676 to training and six to monitoring. Shared task identities can cross this split, so monitoring loss does not measure unseen-task transfer. The training partition contains 21,420,756 total and 11,352,126 supervised tokens under the native template. All training histories fit within the supported context without truncation or removal.

Table 11: Full SFT configuration for the two evaluated models.

Item	Setting
Initialization	Separate original 27B and 35B-A3B base checkpoints (Qwen Team, 2026a;b)
Tuning	Full language-model parameters
Objective	Assistant-token cross entropy, full conversation context
Epochs	8
Learning rate	10^{-6}, cosine decay to 10^{-7} (Loshchilov & Hutter, 2017)
Warmup	0.03
Adam coefficients	$(0.9, 0.95)$ (Kingma & Ba, 2017)
Weight decay; clipping	5×10^{-5}; 1.0
Precision	bfloat16
Per-device batch; accumulation	1; 1
Maximum sequence length	131,072 tokens
Packing lengths (27B / 35B)	131,072 / 98,304
Sequence parallelism (27B / 35B)	4 / 2 (Korthikanti et al., 2023)
Memory strategy	ZeRO-3, optimizer offload, gradient checkpointing (Rajbhandari et al., 2020; Ren et al., 2021; Chen et al., 2016)
Attention	FlashAttention 2 (Dao, 2024)
Seed; data seed	42; 42

The 269 supplemental histories comprise the probe and supplementary collections listed above. Earlier classification data contained a row-order signal; task delivery now shuffles rows and remaps identifiers, and content review checks trajectories for use of this shortcut. Public inputs and private grading files are logically separated. The screening procedures and data provenance are retained with the training export.

Full SFT uses the final checkpoints at steps 664 (27B) and 440 (35B-A3B). Automatic screening yields 422 candidates; final review retains 413 main-collection histories. The internal evaluation cohort contains 30 tasks excluded from training.

E A PUBLIC-FEEDBACK EXAMPLE

One 27B trajectory completes a binary task in 694.2 seconds, obtaining test AUC 0.774371 above gold 0.766243. This task uses the strong anchor, with $c_s = c_o = 0.779811$ and $H = 0$. It illustrates accepted interaction content, rather than closure of a positive structural gap.

Table 12: Selected events from a complete trajectory. Indices are zero-based raw-log positions. Validation quantities retain the programs' original definitions.

Event	Observation or experiment	Subsequent action or output
20	Date and string parsing; initial gradient boosting mean CV AUC 0.7500 (Friedman, 2001).	Agent proposes revised features and tuning.
25	Constructor rejects `subsample`.	Subsequent execution removes the invalid setup.
32	Revised gradient boosting mean CV AUC 0.7674; pooled OOF AUC 0.7672.	Agent explores more configurations.
40–41	Larger search reaches a command timeout.	Agent narrows search and simplifies the ensemble.
48	Gradient boosting plus ExtraTrees reaches pooled OOF AUC 0.7725 (Friedman, 2001).	Agent writes the selected pair's predictions.
51–52	Checks 1,952 rows, columns, finite predictions, and matching IDs.	Final assistant reports completion.

Mean fold AUC and pooled OOF AUC are different statistics; they are not a harmonized improvement curve. Test scores were unavailable during these decisions. The tool-command timeout occurred within an episode that completed before its overall limit. The SFT screen checks the episode-level timeout flag. Full-history supervision preserves this mixture of successful actions and recoverable mistakes.

F EVALUATION PROTOCOLS AND REPORTING

Evaluation overview. Table 2 reports external transfer for the Base and Full SFT models. The remaining experiments examine task calibration, difficulty controls, demonstration yield, trajectory content, and continuous experimentation. Each comparison uses a common evaluation protocol across its model conditions; task-specific metrics retain their native units.

MLE-bench evaluates all 75 competitions with three independent runs per task and weight, following the standard repetition protocol of Chan et al. (2025). Each run has a 24-hour limit and temperature 0.5. Model conditions for a given task and repeat share the same GPU type and CPU/memory limits. Medal rates aggregate repeated runs without best-repeat selection.

SWE-fficiency evaluates all 498 tasks across nine repositories with one agent episode and one final patch per task and weight, following the pass@1 protocol of Ma et al. (2026a). Each episode has a three-hour limit, 100 actions, and a 30-minute step timeout. The execution budget is four vCPUs and 16 GB per worker, with temperature zero. Gold and model patches share the same environment and correctness checks. For each task, normalized speedup is model speedup divided by gold speedup; SR is the harmonic mean of these ratios. An empty patch or correctness failure receives model speedup one, hence normalized speedup $1/$gold speedup. Infrastructure errors and invalid gold runs are tracked as missing evaluations.

Success counts and aggregation. Table 13 reports the numerators and effective denominators underlying the binary rates in Table 2. All four weights use the same task sets: 13 MLAgentBench tasks with eight repeats, 75 MLE-bench tasks with three repeats, and 498 SWE-fficiency tasks with one run. The reported comparisons contain 104, 225, and 498 evaluated outcomes per weight, respectively, with zero missing evaluations or excluded tasks. A completed unsuccessful attempt remains in the denominator. For SWE-fficiency, the common 498-task set has valid gold evaluations for all four conditions.

Table 13: **Success counts and effective denominators for external binary metrics.** Entries are successes / evaluated outcomes. MLE-bench counts runs attaining any medal; SWE-fficiency counts patches satisfying correctness checks. Each benchmark uses the same complete evaluation set across weights.

Model	Weights	MLAgentBench	MLE-bench	SWE correctness
27B	Base	79 / 104	64 / 225	447 / 498
	Full SFT	92 / 104	99 / 225	463 / 498
35B-A3B	Base	59 / 104	70 / 225	413 / 498
	Full SFT	74 / 104	103 / 225	444 / 498

For T tasks, R repeats per task, and binary outcome $y_{tr} \in \{0, 1\}$, the reported percentage and paired gain are

$$p = \frac{100}{TR} \sum_{t=1}^{T} \sum_{r=1}^{R} y_{tr}, \qquad \Delta = \frac{100}{TR} \sum_{t=1}^{T} \sum_{r=1}^{R} (y_{tr}^{\mathrm{SFT}} - y_{tr}^{\mathrm{Base}}). \tag{10}$$

Each task–repeat outcome has equal weight, with no best-repeat selection. We round only the final percentages and gains to two decimals. A displayed gain can therefore differ by 0.01 pp from subtracting the displayed percentages. For example, the 35B-A3B SWE-fficiency correctness gain is $100(444-413)/498 = 6.2249\ldots$ pp, reported as 6.22 pp. DSBench RPG and SWE-fficiency SR retain their continuous benchmark-specific aggregation rules (Jing et al., 2025; Ma et al., 2026a).

For the reported task-bootstrap intervals, repeats are first averaged within task, then tasks are resampled jointly across paired conditions for 10,000 draws with seed 42 (Efron, 1979). Success-rate differences are expressed in percentage points. With one training seed, these intervals quantify task-sampling uncertainty for the evaluated checkpoints.

F.1 DSBench, MLAgentBench, and internal evaluation

DSBench Data Modeling (Jing et al., 2025) evaluates 74 tasks with one run per weight, giving 296 runs across the four Base/Full SFT weights. Each run uses a 30-minute limit, 60 actions, temperature 0.7, and four CPUs with 14 GiB memory. RPG measures nonnegative progress relative to each task's baseline and target, without clipping at one. Invalid submissions and model budget failures receive zero. Model inference resources are allocated separately from executor resources.

MLAgentBench evaluates all 13 tasks with eight independent runs per task and weight, following Huang et al. (2024). Success requires at least a 10% improvement over the starter-model metric. Each run uses a five-hour limit, 50 actions, temperature 0.7, eight CPUs, 56 GiB memory, and one execution GPU. The same task interface, scoring rule, and resource budget apply to the Base and Full SFT conditions. Table 2 gives the aggregate success rates.

Table 14: Paired changes on internal tasks. Repeats are averaged within task; intervals reflect task-sampling uncertainty for the evaluated checkpoints.

Model	Benchmark	SFT − Base	Paired 95% interval
27B	Internal gold (pp)	-1.67	$[-5.00, 0.00]$
35B-A3B	Internal gold (pp)	$+1.67$	$[0.00, 5.00]$

F.2 Internal prediction tasks

The internal cohort contains 30 tasks excluded from training, ten per prediction type. The four model conditions are Base and Full SFT for each model size. Each condition completes two runs per task, giving 240 episodes. Evaluation uses temperature 0.7, top-p 0.95, enabled thinking, 1,800 seconds, and 60 turns for every condition. Repeats are averaged within task for paired comparisons.

F.3 Continuous versus independent execution

The feedback study uses the first four tasks per type from the frozen 30-task internal cohort and four original Base/Full SFT weights, giving 48 paired groups and 144 episodes. Each group con-

tains one 1,800-second/60-turn episode and two fresh 900-second/30-turn episodes, with the same environment and sampling. A seed-42 public-validation split removes 20% of training rows, stratified for classification and random for regression. The agent accesses validation features and scalar scores; targets remain with the trusted grader. The two independent final submissions are selected by final public score: valid beats missing, and ties select the first. Selection is immutable before hidden scoring. The continuous condition submits its final artifact, without hidden-score selection over history.

$$\Delta_{\text{cont}} = (p^{30}_{\text{SFT}} - p^{2\times15}_{\text{SFT}}) - (p^{30}_{\text{Base}} - p^{2\times15}_{\text{Base}}). \tag{11}$$

All four weight/mode observations for a model follow the same task during bootstrap. Equation (11) reports the change in endpoint success-rate differences, in percentage points. Both execution modes use feedback and share cumulative time and turn limits; actual token costs can differ. The study evaluates the 12-task subset with the public-validation protocol above and reports final-submission outcomes.

Table 15: Feedback-study success rates and paired 95% task-bootstrap intervals. Rates are percentages; changes in continuity advantage are in pp. Each endpoint uses 12 tasks.

Model	Weights	Continuous 30 min	Independent 2×15 min
27B	Base	16.67 [0.00, 41.67]	8.33 [0.00, 25.00]
27B	Full SFT	16.67 [0.00, 41.67]	0.00 [0.00, 0.00]
35B-A3B	Base	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]
35B-A3B	Full SFT	0.00 [0.00, 0.00]	8.33 [0.00, 25.00]

	27B	35B-A3B
Change in continuity advantage (pp)	8.33 [0.00, 25.00]	-8.33 [-25.00, 0.00]

F.4 Mechanism settings and descriptive difficulty profiles

Figure 4 reports the main collection's gold/requested rates for 27B Base and 35B-A3B Base. Table 16 gives the committed parameter values and exact task counts. Every task has ten requested episodes per model, each with 1,800 seconds and 60 CLI turns. Thus a cell with n tasks contributes $10n$ requested episodes per model, including blocked and unscored requests. These groups jointly cover all 300 tasks and all 6,000 requested episodes.

Table 16: **Task counts by generating mechanism and parameter.** Numeric settings are committed frequency ω or parity order k. Binary, multiclass, and regression tasks use ROC AUC, macro-F1, and RMSE, respectively. Statistical and representation controls vary within each group.

Family	Setting	Binary	Multiclass	Regression	Total
Polynomial	—	10	14	14	38
Saturating	—	9	6	4	19
Threshold	—	9	11	9	29
Parity	$k = 4$	22	0	0	22
Parity	$k = 5$	20	0	0	20
Periodic	$\omega = 4$	2	13	15	30
Periodic	$\omega = 5$	8	13	17	38
Periodic	$\omega = 6$	4	19	21	44
Radial	$\omega = 4$	9	4	8	21
Radial	$\omega = 5$	3	9	5	17
Radial	$\omega = 6$	4	11	7	22
Total		100	100	100	300

Periodic and radial settings use frequencies $\omega \in \{4, 5, 6\}$; the radial latent function applies 0.8ω to the radius as defined in Appendix A.2. Parity tasks use orders four and five. Stage B sampling intensity is 0.5, with operator composition sampled for each task. The DGP and admission rules

determine the final sample size, noise, and feature settings. The grouped rates therefore characterize corpus coverage and agent outcomes under the committed configurations.

F.5 TRAJECTORY-LENGTH ACCOUNTING

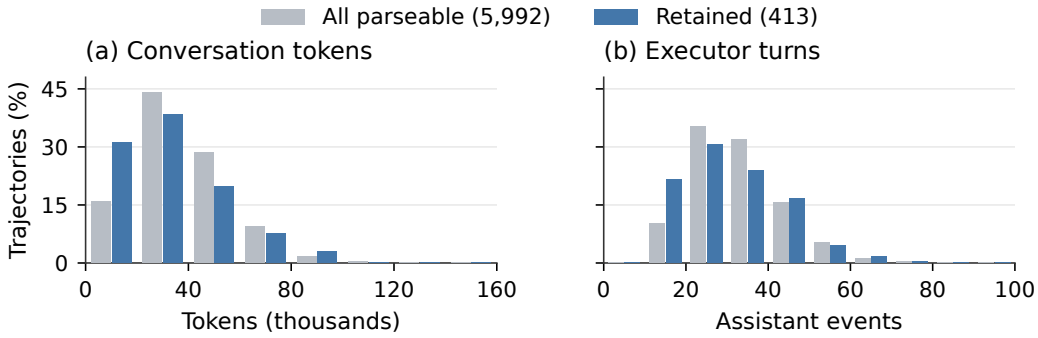


Figure 8: **Collected and retained trajectory lengths.** Histograms use common bins, normalized to 100% within each cohort. Tokens encode each conversation once; executor turns count assistant events. Eight unparseable records are excluded; all 413 retained histories are included.

Length statistics cover the main-collection rollout manifest and the 413 reviewed main-collection histories (305 generated by 27B and 108 by 35B-A3B). The histograms compare these two cohorts; generating-model breakdowns are supplementary, irrespective of which model later trains on a history. A fixed 27B Base tokenizer and native template count the actual content of each complete conversation once, excluding the repeated API transmission of prior context. Original executor turns are audited separately from tool messages or multiple tools in one turn. Records without a valid correspondence are missing; absent or partial trajectories are not assigned zero length.

The audit contains 5,992 parseable trajectories out of 6,000 requests. Five lack a prompt, two have truncated JSONL, and one requires an unsupported nested-conversation conversion. All 413 retained histories are countable. The tokenizer and template are fixed across generating models; executor turns count assistant JSONL events, which may include fragments from a single response. Tool calls and API responses are separate quantities. The retained pool is a subset of all histories, so the distribution comparison is descriptive. The histograms use bins of 20,000 tokens and ten assistant events, normalized within each cohort. Figure 9 retains the full ECDFs, with a logarithmic token axis.

The length distributions describe the main collection and its 413 retained histories. Appendix D gives the composition of the training set.

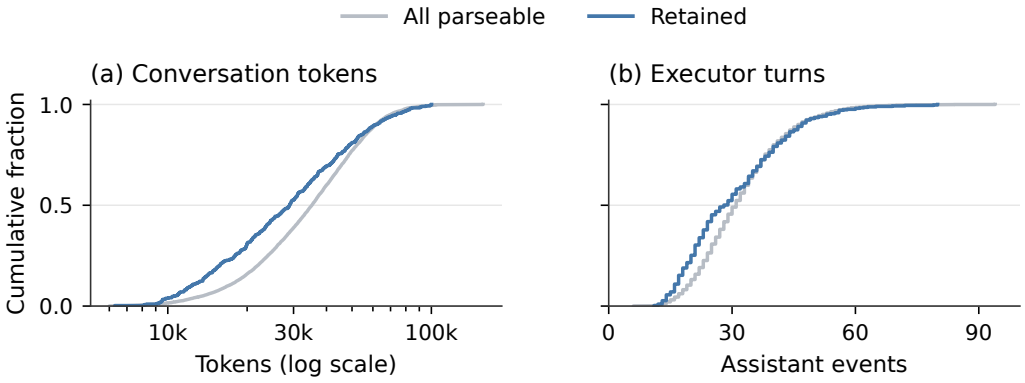


Figure 9: Full cumulative length distributions for the two cohorts in Figure 8.

Table 17: Trajectory-length summaries for the main collection. Token counts encode each conversation once. Executor events and tool calls are counted separately.

		Conversation tokens		Executor turns		Tool calls	
Cohort	n	Median	P90	Median	P90	Median	P90
All parseable	5992	35,202.5	61,523.0	31.0	46.0	22.0	36.0
Retained	413	29,006.0	61,237.6	28.0	47.0	18.0	34.0
All: 27B	2996	28,949.5	51,015.0	31.0	45.0	21.0	33.0
All: 35B-A3B	2996	43,061.0	67,795.5	31.0	47.0	24.0	38.0
Retained: 27B	305	24,177.0	51,542.6	26.0	44.0	16.0	31.0
Retained: 35B-A3B	108	46,679.0	81,766.3	35.5	55.3	25.0	39.0