READING THE TEA LEAVES, AGAIN: CAN LARGE LANGUAGE MODELS PREDICT LANGUAGE MODEL TRAINING RUNS?

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agents now propose research ideas and run the experiments that test them. However, they run every experiment to its end, whereas a researcher babysits each run, reading its configuration and the beginning of the curve, and stops those that show no promise. We ask whether LLMs can read a run the same way and build TEA LEAVES to measure it, from 682 real training runs across pretraining, fine-tuning, and reinforcement learning (RL). A forecaster sees the first 20% of a run with its configuration, description, and code, and predicts the rest, which requires both curve extrapolation and a working knowledge of training dynamics. Across 11 LLMs and a range of forecasting baselines, the strongest LLMs match or exceed the best curve fitter on smooth loss curves (skill 0.66 against 0.62) and lead on noisy RL reward curves (0.43 against 0.13), but their skill falls as the runs get harder to read: on sparsely logged community runs they lose most of it, and no method reliably flags a failing run in advance. Our analyses identify what makes a run hard to read: its early curve is too noisy or too sparsely logged to extrapolate from. Knowledge of training dynamics helps where the curve misleads, related runs in the prompt cut the error on loss curves by 39%, and tools and finer input windows add little for the strongest readers. The benchmark and its database of runs will be released to test LLMs' understanding of training and help research agents decide which runs deserve their compute.

1 INTRODUCTION

Large language model (LLM) agents now propose research ideas and pursue them on their own (Lu et al., 2024; Yamada et al., 2025; Jansen et al., 2025; Schmidgall et al., 2025). Because every idea is expensive to execute, prior work evaluates and trains LLMs to predict the outcome of research from its description: whether an idea will work, which of two proposals will score higher, or what a study will find (Wen et al., 2025; Luo et al., 2025b; Park et al., 2025). Yet even a promising idea has to be executed, and in machine learning (ML) that means training models, run after run, until the idea is judged (Si et al., 2025; 2026a;b). Today's agents, and the benchmarks that score them (Huang et al., 2024; Chan et al., 2025; Nathani et al., 2025), train every run to its end, whereas a researcher babysits a run (Karpathy, 2016): they read its configuration and code, watch the first stretch of its curve, and stop it early if it shows no promise, saving the compute for the next one. Without this judgment, an agent pays for every run in full.

To test whether LLMs can read the same signs,[1] we build TEA LEAVES from 682 real public training runs, and three choices in its design decide what it can measure (§3). Figure 1 illustrates the task. First, a forecaster sees the first 20% of a run together with the *semantics* of the experiment, its configuration, description, and training code, and predicts the rest with uncertainty. Answering well takes both extrapolation and a working knowledge of training dynamics, how the learning rate, model size, data, and algorithm shape a curve (Kaplan et al., 2020; Tissue et al., 2025; Luo et al., 2025a); two companion tasks isolate each. Second, the runs span pretraining (PT), fine-tuning (FT),

[1]The title borrows from Chang et al. (2009), who asked whether humans can read a topic model from the faint signs it leaves. We ask, again, whether a model can read a training run from its faint signs, a partial curve, a configuration, and code, with GPU hours riding on the answer.

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.

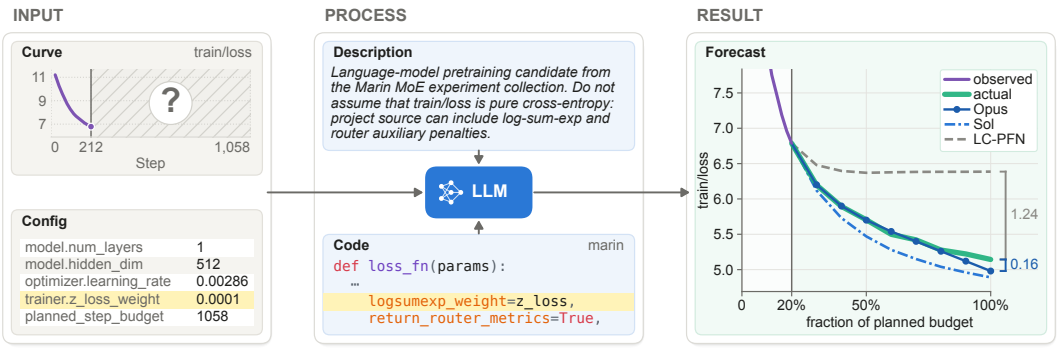

Figure 1: The task on one test run (Marin MoE pretraining). Input: the first 20% of the loss curve, the rest hidden, and the configuration. Process: an LLM reads them with the description and code, which here flag auxiliary loss terms (highlighted). Result: its forecast against the actual curve (green), with GPT-5.6 Sol and LC-PFN; brackets mark the final-loss error.

and reinforcement learning (RL), whose curves range from smooth losses to noisy rewards, so we can see where extrapolation stops sufficing. Third, the runs keep their provenance. Curated runs come from controlled sweeps with recorded lineage, so related runs can be retrieved and leakage controlled; community runs from the Hugging Face Hub pose the question in the wild, and failed runs test whether a failure is visible in advance. A skill score S normalizes each forecast by the error of repeating the last observed value, so runs can be compared across metrics and scales.

We evaluate 11 LLMs against 8 baselines, spanning curve fitters, time-series foundation models, and predictors trained on past runs (Adriaensen et al., 2023; Ansari et al., 2025; Das et al., 2024; Zhang et al., 2026), and find that **the strongest LLMs are competitive on loss curves and lead on rewards, yet their skill falls as the runs get harder to read** (§4.1). Smooth loss curves are the easy case: the strongest LLMs match or exceed the best curve fitter, which knows nothing about the experiment (pooled over pretraining and fine-tuning, GPT-5.6 Sol reaches a skill of 0.66 and Claude Opus 5 0.65, against 0.62 for LC-PFN). Noisy reward curves are harder: the best LLM reaches 0.43, whereas the best curve fitter in our main comparison reaches 0.13 and most methods barely improve on repeating the last value. Sparsely logged community runs are harder still, and every method loses most of its skill there (Claude Opus 5 falls to 0.31). Hardest of all are runs that eventually fail: no method, LLM or otherwise, tells them apart from healthy runs early enough to save compute (§4.2).

We then analyze what separates the easy runs from the hard ones, and what a forecaster needs in order to read the hard ones. The easy runs are those whose *shape already carries most of the answer*: on smooth loss curves, the strongest LLMs reach most of their skill from a tenth of the run, and neither finer input windows (§4.2) nor the semantics of the experiment (§4.1) help consistently. On the hard runs, a forecaster needs information that the curve does not carry, and we find two kinds that help. The first is a **working knowledge of training dynamics**. On reward curves, a third of the strongest LLM's advantage over curve fitters comes from reading the semantics (§4.1). Not every model has this knowledge: asked which of two anonymized settings produced which curve, three of nine models answer at chance, while the five strongest are right over 80% of the time. The knowledge can be taught to some extent: a 9B model fine-tuned on public runs with semantics matches the best LLM on loss curves but still trails on rewards (§4.3). The second is **experience from related runs**. Related runs in the prompt cut the error on loss curves by 39%, because they show how runs like this one went on (§4.4); tools alone, a sandbox and literature search, add little (§4.5). For research agents, this suggests that runs are worth logging densely and storing with their configuration, code, and lineage, and that the expensive call is better spent on runs whose early curve does not already give the answer.

In summary, we contribute (1) TEA LEAVES, a benchmark of 682 real training runs at three stages, with its scorer, every model answer, the database of runs, and the trained readers; (2) evidence that whether a run can be read depends on what its log carries, and that beyond the log only knowledge of training dynamics and related runs help; and (3) a measurement of that knowledge across 11 LLMs, and a recipe that trains a small model to match the best LLM on loss curves. We hope TEA LEAVES lets research agents be evaluated on, and trained for, the judgment a researcher applies before spending compute.

2 Related Work

Predicting training from numbers. Scaling laws predict the final loss from model and data size (Kaplan et al., 2020; Hoffmann et al., 2022) and require a sweep of completed runs; learning-curve extrapolation forecasts from a run's own prefix with parametric families (Domhan et al., 2015), Bayesian models (Swersky et al., 2014), or prior-fitted networks (Adriaensen et al., 2023; Rakotoarison et al., 2024), and multi-fidelity optimization uses such forecasts to stop runs early (Wistuba et al., 2022). Recent predictors condition on more of the run: schedule-aware laws (Tissue et al., 2025; Luo et al., 2025a), NCPL, which maps a numeric configuration to the residual of a scaling-law fit (Zhang et al., 2026), NeuNeu, which forecasts downstream accuracy from training signals (Hu et al., 2026), and compute–performance curves for RL (Khatri et al., 2026). Time-series foundation models (Ansari et al., 2024; 2025; Das et al., 2024) and LLMs prompted with numbers (Gruver et al., 2023; Requeima et al., 2024) forecast generic series, and whether text improves an LLM's numeric forecast is contested (Tan et al., 2024; Williams et al., 2025; Ashok et al., 2026). The generic forecasters read only the series, and the run-specific predictors read numbers rather than text; we test what the configuration, description, and code add to the curve.

Predicting the outcome of research. LLMs have been asked to anticipate results from descriptions: which of two ideas will score higher (Wen et al., 2025), how a neuroscience or social-science study turns out (Luo et al., 2025b; Ashokkumar et al., 2026), what score a method will reach (Park et al., 2025; Sehwag et al., 2026), and what a community will study next (Li et al., 2026; Wang et al., 2026); the same models forecast real-world events at near-human level (Halawi et al., 2024). Theories distilled from the literature predict later results better than a model's parameters alone (Jansen et al., 2026), hypothesis generators learn from past outcomes which ideas will work (Vasu et al., 2025), and LLM surrogates guide Bayesian optimization (Liu et al., 2024). Much of this work predicts research endpoints; our target is the logged training trajectory, published or not.

Agents that run experiments. Benchmarks for research agents score what the agent ships: a model improved on a fixed task (Huang et al., 2024; Chan et al., 2025), a result in a research-level challenge (Nathani et al., 2025), a replication graded against a paper (Starace et al., 2025), or engineering progress measured against human experts under a time budget (Wijk et al., 2025); full systems propose, run, and write up experiments end to end (Lu et al., 2024; Jansen et al., 2025; 2024). Deciding mid-run which experiments to keep is part of every such task but is never measured on its own; Tea Leaves isolates that step. Our database of runs (§4.4) follows retrieval-augmented forecasting (Tire et al., 2026), which retrieves by series shape, whereas we retrieve by configuration; large public curve collections show how varied the shapes are (Yan et al., 2025).

3 The Tea Leaves Benchmark

Tea Leaves measures the forecast behind a researcher's decision to stop or continue a run: the forecaster sees the settings and the prefix and predicts the rest; scoring the stopping decision itself is left to future work. §3.1 defines the task and its metric, §3.2 the data, and §3.3 the forecasters.

3.1 Task formulation

A run is a triple $r = (c, s, y)$: a numeric configuration c (learning rate, batch size, model size, budget, and the other fields the project logs), the semantics s (a description of the task and data, facts about the source project, and excerpts of the training code), and a metric curve $y : [0, 1] \to \mathbb{R}$ indexed by the fraction x of the run's *planned* budget in its own axis (tokens, steps, or epochs). Together, c and s describe the experiment: its settings, task, and implementation. The curve is logged densely, and the observed part is summarized as window means over the first ten windows of width $\delta = 0.02$,

$$\bar{y}_k = \text{mean}\{\, y(x) : x \in ((k-1)\delta,\, k\delta] \,\}, \qquad k = 1, \ldots, 10, \tag{1}$$

so that the prefix $y_{\le 0.2}$ is the ten values $\bar{y}_{1:10}$. Given $(\bar{y}_{1:10}, c, s)$, a forecaster returns the 10%, 50%, and 90% quantiles of y at the query grid $\mathcal{X} = \{0.3, 0.4, \ldots, 1.0\}$; the median $\hat{y}$ is the point forecast. This is the Full condition, in which a researcher or an agent would use a forecaster. Answering well requires two abilities: extrapolating the observed curve, and a working knowledge of training dynamics, how the learning rate, model size, data, and algorithm shape a curve (Kaplan et al., 2020; Tissue et al., 2025; Luo et al., 2025a). For example, a flat prefix during learning-rate warmup need

not imply that training has converged. Two further tasks probe these abilities. Forecasting from the curve alone, as curve fitters do (Domhan et al., 2015; Adriaensen et al., 2023), tests extrapolation by itself; Appendix E.1 removes s and c in turn. A matching task tests the knowledge by itself: given two runs from the same sweep, the model sees both curves and their anonymized settings and says which setting produced which curve (Appendix E.2).

Metric. Curves differ in metric, unit, and scale (a loss in nats, a reward in $[0, 1]$), so errors cannot be averaged across runs; we therefore score each run against its own persistence forecast $\hat{y}_0(x) \equiv \bar{y}_{10}$, which repeats the last observed value. A forecast is scored by its mean absolute error over the grid in the run's own units, and by the ratio of that error to the persistence error:

$$\mathrm{MAE}(r) = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \big|\hat{y}(x) - y(x)\big|, \qquad \rho(r) = \frac{\mathrm{MAE}(r)}{\mathrm{MAE}_0(r)}. \tag{2}$$

The ratio ρ is unit-free, so we aggregate it geometrically over the runs of a cell g (a stage $\times$ source group) and then over cells,

$$R = \exp\Big(\frac{1}{|G|} \sum_{g \in G} \frac{1}{|g|} \sum_{r \in g} \ln \rho(r)\Big), \qquad S = 1 - R, \tag{3}$$

and report the *skill score* S, as in weather forecasting (Murphy, 1988). S is 0 for repeating the last value, 1 for a perfect forecast, and negative for a forecast worse than persistence; 95% intervals resample lineage components, so that runs sharing a trajectory are never resampled independently (worked example and edge cases in Appendix C). The main comparison scores medians; Appendix C compares quantile losses, and Appendix G examines the trained reader's interval coverage.

3.2 Data

Sources. We enumerate runs from three kinds of source: public experiment-tracker projects of open training efforts (19,650 run identities recorded in 236 collection attempts), training-state files on the Hugging Face Hub [2] (47,929 listed models, 14,370 with a trainer state, 4,503 parsed), and a public speedrun log archive (1,516 logs) [3]. Of these, 17,805 runs have usable numeric curves, and every count below starts from them (sources and funnel in Appendix B).

Filtering. A run enters the benchmark only if it passes every one of the following rules. (1) **Defined target.** The run logs a training metric with a known axis and a known planned budget, so that y and the fraction x are defined (6,780 of 17,805 runs). (2) **Visible semantics.** The project exposes a configuration and either code or a description, so that s and c can be shown; we keep the configuration as logged and provide source-linked code excerpts. (3) **Curve quality.** The log has enough points before and after the 20% mark to define every window mean and every query point, no duplicated history, and at most two seed replicates of the same setting (counts in Appendix B). (4) **Controlled variation (test sets only).** The run belongs to a sweep that varies a setting a researcher changes (learning rate, batch size, warmup, budget, model size, data, or initial checkpoint), and every stage draws on several sources.

Splitting. Test sets are drawn first: 150 pretraining and 150 fine-tuning runs, drawn together with the matching pairs from 589 run identities, and 150 RL runs from more than twenty repositories. Runs that resume from or branch off one another share part of a trajectory, so whole lineage components are assigned to one split. Every run also carries its sweep family and source project; training pools remove every test sweep family (1,463 runs remain) or, where stated, the whole source (§4.3). The test sets were frozen with content hashes before any model was evaluated. Table 1 summarizes the four test sets, and Figure 7 in Appendix B shows one run of each as logged. The in-the-wild set holds 232 community runs by 184 authors, drawn from 1,513 Hub candidates; none shares a repository with another set, and we removed 36 candidates whose author or token collided with the frozen sets.

Setup. Following NeuNeu (Hu et al., 2026), the forecaster observes the first 20% of the planned budget and predicts the rest (§4.2 varies both); an LLM receives the ten prefix means (on RL, the released prefix, which is denser; Appendix A.2), the configuration, the description, and the code excerpts in one prompt and answers once (decoding and exact model identifiers in Appendix A.1).

[2] `https://huggingface.co/docs/hub/index`

[3] `https://github.com/KellerJordan/modded-nanogpt`

Table 1: The TEA LEAVES test sets; *Ex.* gives one question from each set as the forecaster sees it, with the settings that vary within its sweep. Sources: Marin (Marin Community, 2025), whose fine-tuning runs follow the Tülu 3 recipe (Lambert et al., 2024), DCLM (Li et al., 2024), OLMoE (Muennighoff et al., 2024), and Open-R1 (Hugging Face, 2025).

Stage	**Metric (axis)**	**Sources**	**Runs**	**What varies within a sweep** (underline)
Pretraining	loss (tokens)	Marin, DCLM, OLMoE	150	learning rate, model size, batch, budget, data
Ex.: a Marin run of width 1280 at learning rate 1.6, 20% of its 30.8B tokens observed; *where does train/loss end?*				
Fine-tuning	loss (steps, epochs)	Marin (Tülu 3 recipe), Open-R1	150	learning rate, epochs, data mix, checkpoint
Ex.: Open-R1 fine-tuning of a 7B checkpoint for three epochs, the first epoch partly observed; *where does the loss settle?*				
RL	reward (steps)	20+ lab and paper repositories	150	reward, budget, checkpoint, algorithm
Ex.: GRPO on math prompts from a 1.5B checkpoint, 86 of 430 steps observed; *will the mean reward keep rising?*				
In the wild	mixed	Hugging Face Hub, 184 authors	232	uncontrolled
Ex.: a community LoRA run whose trainer state logs 62 points, twelve of them in the prefix; the repository is the only description.				

Unless stated otherwise, every experiment in §4 uses this setup; one complete question and its answer are shown in Appendix J.1.

3.3 BASELINES AND MODELS

We evaluate five families of forecaster, each standing for a way a researcher or a lab could obtain a forecast today. (1) **Curve fitting** extrapolates the prefix alone, as a researcher would first: the last value, a power-law fit, an AIC-weighted ensemble of parametric fits, and the prior-fitted networks LC-PFN (Adriaensen et al., 2023) and FT-PFN (Rakotoarison et al., 2024). (2) **Time-series foundation models** forecast any numeric series zero-shot: Chronos-2 (Ansari et al., 2025) and TimesFM-2.5 (Das et al., 2024; Google Research, 2025). (3) **Predictors trained on past runs** represent a lab that learns from its own history: NeuNeu-NoLoss (Hu et al., 2026) and NCPL (Zhang et al., 2026), a predictor built for neural scaling laws, retrained with the authors' code on our pool with every test sweep held out (implementation of every baseline in Appendix A.2); two shape regressors trained on the pool are reported in Appendix D.8. (4) **Closed-weight LLMs** read the curve and the semantics together: GPT-5.6 Sol and GPT-5.6 Luna (OpenAI, 2026), Claude Opus 5 and Claude Haiku 4.5 (Anthropic, 2026; 2025), and Gemini 3.1 Pro and Gemini 3.8 Flash (Google DeepMind, 2026b;a). (5) **Open-weight LLMs**, which a lab can run and train itself: Kimi K3 (Kimi Team, 2026), Qwen3-Next-80B (Qwen Team, 2025), DeepSeek-V3.2 (DeepSeek-AI, 2025), Muse-Glimmer-30B (Meta Superintelligence Lab, 2026), and Qwen3.5 at 0.8B, 2B, 4B, and 9B (Qwen Team, 2026). All LLMs are zero-shot with the same prompt unless otherwise specified.

4 EXPERIMENTS AND ANALYSIS

We first report the benchmark (§4.1) and then four analyses that each narrow the scope to one question: how much of a run a reader must see (§4.2), whether a small model can be trained to forecast (§4.3), whether retrieving past runs helps (§4.4), and whether tools make up for what the prompt lacks (§4.5). Figure 2 compares the 11 LLMs with the forecasting baselines; three smaller Qwen3.5 models appear only in Appendix D.1. Unless stated otherwise, every forecaster sees the first 20% of a run, LLMs see the full semantics, and scores are the skill score S of §3.1 with 95% intervals over lineage components; paired contrasts are collected in Appendix C.2.

4.1 READING THE TEA LEAVES

The strongest LLMs are competitive on loss curves and lead on rewards. Figure 2 plots forecasting skill on the four test sets, one column per stage; several patterns stand out. On pretraining runs Claude Opus 5 reaches $S = 0.70$, GPT-5.6 Luna 0.68, and GPT-5.6 Sol 0.66, against 0.66 for LC-PFN, the best curve fitter, and at most 0.46 for every other zero-shot baseline (Appendix D.1). On fine-tuning runs the best LLMs reach 0.63–0.65 against 0.55 for LC-PFN. Predictors trained on past runs with the test sweep held out do not close this gap (NCPL 0.43 and -0.04; NeuNeu-NoLoss 0.35 and 0.15); a predictor that has seen the test run's own sweep interpolates within it (retrained

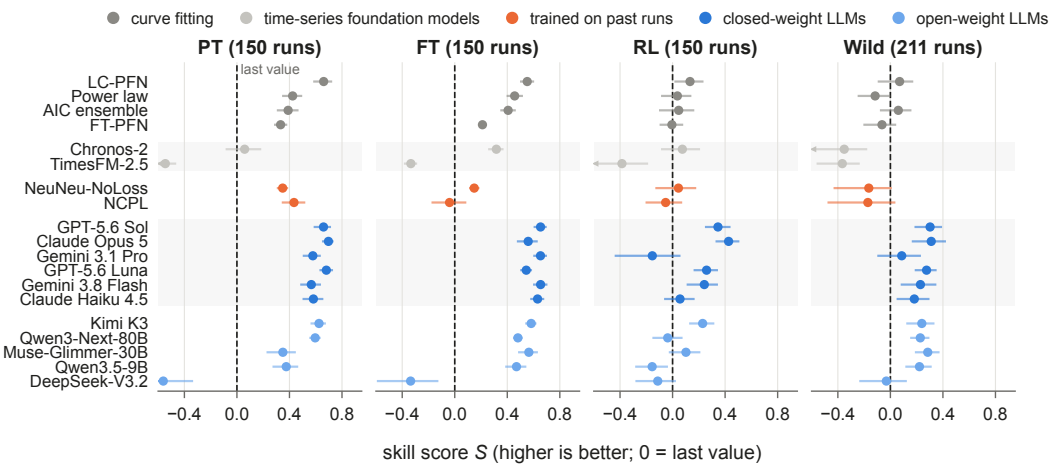

Figure 2: Forecasting skill S on PT, FT, RL (150 runs each), and the 211 community runs that every model evaluated first on this set answered validly (Appendix D.7); each model is scored on its own valid answers, and RL on the 133 runs whose logged series moves after the prefix (Appendix C). Bars: 95% intervals; dashed: last value. Pool-trained models hold out the test sweep.

NCPL falls from 0.79 to 0.28 when that sweep is held out; Appendix D.8), and the LLMs' gain over LC-PFN concentrates on the runs where the fitter errs (Appendix D.4). Within the LLMs, the six closed-weight models fall within 0.13 of the best on loss curves, and the open-weight models vary widely (Muse-Glimmer-30B 0.35 and 0.56).

Reward curves are harder, and skill on losses does not transfer. Across the columns, the spread between stages exceeds the spread between most models within a stage. Reward curves are noisy enough that the last value is a strong forecast, so on RL runs the best curve fitter in Figure 2 reaches 0.13, Claude Opus 5 0.43, GPT-5.6 Sol 0.35, and no other method in the figure exceeds 0.26. RL is not simply unforecastable: the closed models spread from -0.15 to 0.43, and Gemini 3.1 Pro, level with GPT-5.6 Sol on fine-tuning (0.65), scores -0.15 (per-project RL results in Appendix D.6).

In the wild, every method loses most of its skill with sparse, noisy logs. In the last column, on the 211 community runs, Claude Opus 5 falls to 0.31, GPT-5.6 Sol to 0.30, LC-PFN to 0.07, and NCPL, retrained on our curated runs, to -0.17; the drop concentrates in RL runs and in runs by individual authors. Every baseline without an LLM is at or near zero here, so the skill that remains in the wild comes from reading the run rather than from its curve. The obvious explanation, that community runs are poorly described, does not account for it: removing every identifying field, or adding everything the model cards recover, leaves the error ratio within 0.96–1.02 of the original. The logs explain much of it: thinning curated curves to the community density (7 raw points per window against 59) with comparable noise reproduces, in the error floor of a hindsight-best fit, 95% of the gap (Appendix D.7).

Semantics help interpret the prefix. Adding semantics raises Claude Opus 5's RL skill from 0.33 to 0.43, a third of its lead over the best curve fitter (Table 10), and cuts its error by about a third on warmup-dominated prefixes; on loss curves the effect is mixed, from -0.19 for Claude Opus 5 on fine-tuning to $+0.11$ for GPT-5.6 Luna on pretraining (Appendices C.2 and E.1). The matching task shows the knowledge and its limit: the strongest models match settings to completed curves with 82–86% accuracy, yet assignments derived from their forecasts are near chance (Appendix E.2).

4.2 How much of a run must a reader see?

The main experiments observe 20% of each run, average the logs in 2% windows, and score a run only if it finishes. We vary the first two on fixed subsets (100 and 86 questions); for the third, we ask whether 100 failed runs can be told from 200 matched healthy runs before they fail; protocols and full tables are in Appendix F.

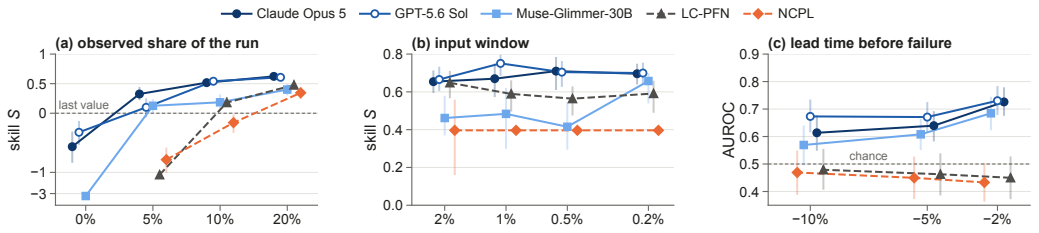

Figure 3: (a) Skill S against the observed share of the run, relative to the 20% last value throughout (at 0% only the configuration is given; y axis compressed below -1); (b) skill against the input averaging window; (c) failure-detection AUROC against lead time. Orange: retrained NCPL, whose ten-window score is repeated in (b); bars: 95% intervals.

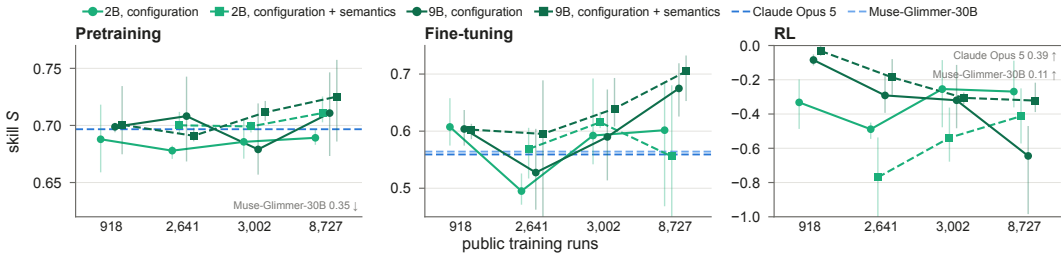

Figure 4: Reader skill S by stage, backbone, and training-pool size, with the configuration alone or with the semantics; dashed: Claude Opus 5 and Muse-Glimmer-30B on the same runs; faint bars: range over three seeds.

A tenth of a run carries most of the signal. Figure 3a shows skill as the observed share grows. Claude Opus 5 and GPT-5.6 Sol reach 0.52–0.54 from 10% of the budget and 0.61–0.62 at 20%. LC-PFN and the retrained NCPL are negative at 5% (-1.13, -0.79), still at 0.18 and -0.16 at 10%, and need the full 20% to reach 0.48 and 0.35.

Finer windows help Muse-Glimmer-30B, while the strongest readers remain stable. Narrowing the input window from 2% to 0.2% leaves Claude Opus 5 and GPT-5.6 Sol at 0.65–0.75 but raises Muse-Glimmer-30B from 0.46 to 0.66 (Figure 3b). This differs from sparse community logging: finer windows expose more of an existing trace, whereas missing measurements cannot be recovered by changing the window.

No monitor catches a failing run early. Figure 3c turns to failure. Two percent of the budget before a failure, the LLMs separate failing from healthy runs with AUROC 0.73 (Claude Opus 5 and GPT-5.6 Sol: three times in four they rank a failing run above a healthy one; 0.68 for Muse-Glimmer-30B), which fades to 0.61–0.67 ten percent ahead; LC-PFN and NCPL used as monitors stay at chance (0.43–0.48). Run online at checkpoints with one false alarm allowed per five healthy runs, no monitor catches half of the failures (the best 0.41, interval up to 0.51, at the cost of the healthy runs it stops), and LLMs with the full context do no better than boosted trees on curve statistics (0.36 against 0.41; both protocols in Appendix F). The scalar log warns late.

4.3 Can we train a reader?

Open-weight models vary widely on loss curves (§4.1), and a lab may want an in-house reader for its own runs. We ask how much of the gap training on public runs closes: we fine-tune Qwen3.5-2B and 9B as text-to-text regressors (Song et al., 2024; Akhauri et al., 2025) that read the prefix and the configuration, either alone or with the semantics of §3.1 condensed into one paragraph, and predict each query point as a ratio to the last observed value; we call the result TeaReader. Training pools grow from the 918-run reference pool to 2,641, 3,002, and 8,727 runs by adding public pretraining sweeps, Hugging Face RL runs, and further public pretraining sweeps, admitted under the rules of §3.2 except for 22 records in the two middle pools later found related to test runs; each reader trains for 1,000 updates with three seeds, with a recipe fixed on the reference pool (Appendix G).

Training closes the gap on loss curves and not on rewards. Figure 4 plots skill against training runs, one panel per stage. Before training, the 2B backbone scores 0.58 (pretraining) and 0.18 (fine-tuning) with the curve and configuration, and the 9B backbone 0.38 and 0.47 with the full prompt. Trained, the 2B reader reaches 0.69 and 0.61 on pretraining and fine-tuning, level with Claude Opus 5 on the same runs, and the 9B reader with the semantics on the largest pool reaches 0.73 and 0.71, level with the best LLM. The semantics raise the 9B reader's average over the three stages on every pool (by 0.02 to 0.12), mostly on fine-tuning and RL, and hurt the 2B reader on RL. On RL no reader beats the last value, and more public runs make it worse: the 9B reader falls from -0.08 on the reference pool to -0.64 on the largest, and the semantics only slow the fall (-0.03 to -0.32), so the best three-stage average (0.42) stays far below Claude Opus 5 (0.55).

4.4 Can past runs give a reader a memory?

Every reader so far sees only the run in front of it, yet a lab keeps its past runs and the community keeps many more in public. We therefore let the reader retrieve related runs from a database of past runs stored with their configuration, code, and lineage. A shared community collection could extend this memory beyond one organization, grown by researcher submissions of curves and their context.

On 125 questions we embed the configuration, fetch the three nearest runs outside the question's sweep family, and add their complete curves to GPT-5.6 Sol's prompt; databases of 17 to 1,726 runs are subsamples of the same public pool (Appendix H).

A bigger database helps because it holds a closer run. The smallest database barely helps (10%, interval including zero) and the largest cuts GPT-5.6 Sol's error by 39%, as the nearest run becomes more similar (Figure 5); questions for which a sibling series was retrieved gain about as much as the rest (42% against 39%), so sibling leakage does not explain the gain. Retrieval also matches the benefit of training: with the sweep excluded, GPT-5.6 Sol with retrieval beats the best regressor trained on the same runs in four of five sources; with the question's own sweep family allowed, it beats the regressor trained on it in all four sources that have one (Marin pretraining MAE 0.108 against 0.209); and it picks the best change in a real sweep 61% of the time with retrieved sweeps against 46% without (Appendix H).

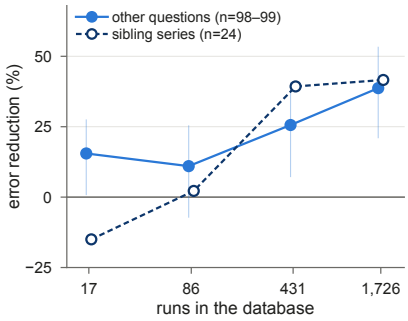

Figure 5: GPT-5.6 Sol's error reduction with three retrieved runs, against database size. Hollow: questions for which a run from the same experiment series, never the same sweep, was retrieved.

4.5 Does effort substitute for information?

A run comes with a repository, a log, and related literature; does a forecaster that can open and use them do better than one given the prompt alone? We let Claude Opus 5 and GPT-5.6 Sol work as agents on 100 pretraining, 100 fine-tuning, and 50 RL questions: the same material as a folder, with file access and Python in a sandbox for up to twenty turns; a second variant adds literature and code search with the run's own project filtered out (Appendix I; 6 of 1,000 attempts were not run).

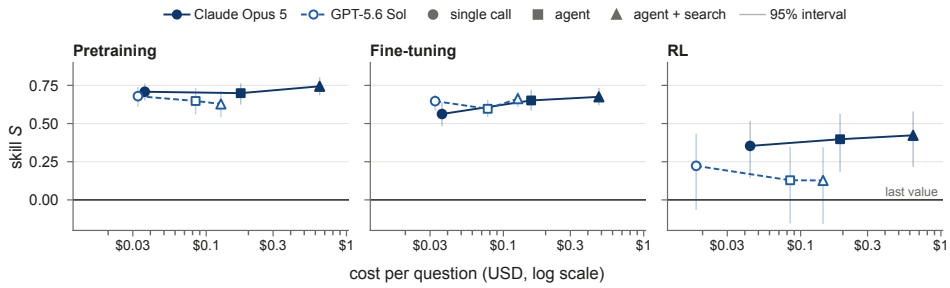

Figure 6: An agent costs 2–18 times a single call and changes skill by at most 0.11. Skill S against cost per question for Claude Opus 5 (filled) and GPT-5.6 Sol (hollow): single call (circle), agent (square), agent with search (triangle); bars: 95% intervals.

Agents work like researchers and forecast like single calls. Figure 6 plots skill against cost. The agents opened the code, fitted the prefix, and searched (pilot logs in Appendix I), yet Claude Opus 5 gained 0.09 $[0.04, 0.14]$ from the sandbox and 0.11 $[0.05, 0.17]$ with search added on fine-tuning only and stayed within 0.07 of the single call elsewhere; GPT-5.6 Sol gained nothing and lost 0.06 on fine-tuning with the sandbox. An agent costs 2–5 times a single call, and 4–18 times with search. In the pilot (Appendix I), an agent given the curves of related runs was level with a single call given the same curves, which alone cut the error by 24% (Claude Opus 5) and 44% (GPT-5.6 Sol). What a reader lacks is not effort but the runs that went before (§4.4, §5).

5 Discussion

Where reading runs out, and what a reader would need next. Reward curves during RL are hard to forecast (Figure 2: 0.43 for the best LLM against 0.13 for the best curve fitter), hard to learn from public runs (Figure 4: every trained reader trails the last value on RL), and a failing run is not caught early enough to save it (Figure 3c). By loose analogy with the halting problem, we call the last of these the *neural halting problem*. Prior work characterizes instabilities (Wortsman et al., 2024; Takase et al., 2025; Molybog et al., 2023; Chowdhery et al., 2023) or warns of divergence from internal activations and reference runs (Huang et al., 2026; Bergsma et al., 2026); our results suggest the scalar log alone carries little warning, and a better one may lie in the data, outputs, gradients, and checkpoints a researcher inspects next. Mounting them in the agent harness of §4.5, for the model to check as a researcher would, would show whether the limit is the log or the reader.

What it means if an LLM can read the tea leaves. A researcher who can predict a run's curve from its settings has compressed the runs they have watched into an expectation; an LLM that does the same has compressed public training reports, logs, and code into its weights, the record of the process that produced it. The leaf reading asks two things: how much of a run is decided by its setup rather than discovered by running it (a right forecast spares compute), and how much of that the field has written down. Our results answer both in part. On loss curves, GPT-5.6 Sol orders a source's runs from the configuration alone and the strongest models reach most of their skill from a tenth of the run, so much of a pretraining or fine-tuning run is predictable from its setup, and the record holds those regularities, though as qualitative knowledge: matching settings to curves is easy, extrapolating from them is not. On reward curves they read far less, and failures barely at all; these are also the outcomes least often written up. Tea Leaves thus measures how much of the practitioner's tacit knowledge has been written down, a boundary that will move as agents publish their own runs.

What a lab can do with Tea Leaves. A lab can train an in-house reader, since under a thousand logged runs and about a GPU-hour give a 2B model that matches Claude Opus 5 on loss curves (§4.3), or keep a database of runs that a frontier model reads at query time, which cuts the error by 39% with the 1,726-run database and no training (§4.4). Both need runs kept with configuration, code, and lineage, as ours are.

Scope and limitations. Every run in Tea Leaves trains a language model and logs a scalar; the task and the scorer apply unchanged to any domain that logs runs, such as vision (Alabdulmohsin et al., 2022), robotics (Lin et al., 2025), and chess (Zhang et al., 2025). The curated runs come mostly from large, well-instrumented projects, so the benchmark likely overstates what forecasters achieve on typical runs; the in-the-wild set shows the direction of that bias. The runs also do not record why their authors ran them, and a prospective test inside a research loop comes next.

6 Conclusion

We built Tea Leaves, 682 real public runs, to test whether LLMs can read a training run from its first 20% and the semantics of the experiment. The strongest LLMs match the best curve fitter on smooth losses and lead on noisy rewards, and every method loses most of its skill on sparse community logs. On loss curves, a small open model trained on public runs matches the best LLM, and related runs in the prompt cut the error by 39%; tools and finer input windows add little, and no monitor catches a failing run early enough to save it. This judgment is thus measurable, partly learnable, and bounded by the log. Benchmark, scorer, model answers, database, and trained readers will be released upon acceptance.

REFERENCES

Steven Adriaensen, Herilalaina Rakotoarison, Samuel Müller, and Frank Hutter. Efficient Bayesian learning curve extrapolation using prior-data fitted networks. In *Advances in Neural Information Processing Systems*, 2023.

Yash Akhauri, Bryan Lewandowski, Cheng-Hsi Lin, Adrian N. Reyes, Grant C. Forbes, Arissa Wongpanich, Bangding Yang, Mohamed S. Abdelfattah, Sagi Perel, and Xingyou Song. Performance prediction for large systems via text-to-text regression. *arXiv preprint arXiv:2506.21718*, 2025.

Ibrahim Alabdulmohsin, Behnam Neyshabur, and Xiaohua Zhai. Revisiting neural scaling laws in language and vision. In *Advances in Neural Information Processing Systems*, 2022.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024.

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, Mononito Goswami, Shubham Kapoor, Danielle C. Maddix, Pablo Guerron, Tony Hu, Junming Yin, Nick Erickson, Prateek Mutalik Desai, Hao Wang, Huzefa Rangwala, George Karypis, Yuyang Wang, and Michael Bohlke-Schneider. Chronos-2: From univariate to universal forecasting. *arXiv preprint arXiv:2510.15821*, 2025.

Anthropic. System card: Claude Haiku 4.5. `https://www.anthropic.com/claude-haiku-4-5-system-card`, 2025. Model card.

Anthropic. System card: Claude Opus 5. `https://www.anthropic.com/claude-opus-5-system-card`, 2026. Model card.

Arjun Ashok, Andrew Robert Williams, Vincent Zhihao Zheng, Irina Rish, Nicolas Chapados, Étienne Marcotte, Valentina Zantedeschi, and Alexandre Drouin. Beyond naïve prompting: Strategies for improved context-aided forecasting with LLMs. *Transactions on Machine Learning Research*, 2026.

Ashwini Ashokkumar, Luke Hewitt, Isaias Ghezae, and Robb Willer. Large language models can predict the results of social science experiments. *Nature*, 656(8126):115–122, 2026.

Shane Bergsma, Bin Claire Zhang, Nolan Dey, Shaheer Muhammad, Gurpreet Gosal, and Joel Hestness. Scaling with collapse: Efficient and predictable training of LLM families. In *International Conference on Learning Representations*, 2026.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *International Conference on Learning Representations*, 2025.

Jonathan Chang, Jordan Boyd-Graber, Sean Gerrish, Chong Wang, and David M. Blei. Reading tea leaves: How humans interpret topic models. In *Advances in Neural Information Processing Systems*, 2009.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark

Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. PaLM: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *International Conference on Machine Learning*, pp. 10148–10167, 2024.

DeepSeek-AI. DeepSeek-V3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *International Joint Conference on Artificial Intelligence*, pp. 3460–3468, 2015.

Google DeepMind. Gemini 3.8 Flash model card. `https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-8-Flash-Model-Card.pdf`, 2026a. Model card.

Google DeepMind. Gemini 3.1 Pro model card. `https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-1-Pro-Model-Card.pdf`, 2026b. Model card.

Google Research. TimesFM 2.5 (timesfm-2.5-200m). `https://huggingface.co/google/timesfm-2.5-200m-pytorch`, 2025.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are zero-shot time series forecasters. In *Advances in Neural Information Processing Systems*, 2023.

Danny Halawi, Fred Zhang, Yueh-Han Chen, and Jacob Steinhardt. Approaching human-level forecasting with language models. In *Advances in Neural Information Processing Systems*, 2024.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems*, 2022.

Michael Y. Hu, Jane Pan, Ayush Rajesh Jhaveri, Nicholas Lourie, and Kyunghyun Cho. Neural neural scaling laws. *arXiv preprint arXiv:2601.19831*, 2026.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating language agents on machine learning experimentation. In *International Conference on Machine Learning*, 2024.

Ruixuan Huang, Hantao Huang, Yifan Huang, Ansheng You, Zhenxing Zhang, and Shuai Wang. Mechanism-driven monitors for preemptive detection of LLM training instability. *arXiv preprint arXiv:2606.28116*, 2026.

Hugging Face. Open R1: A fully open reproduction of DeepSeek-R1. `https://github.com/huggingface/open-r1`, 2025.

Peter Jansen, Marc-Alexandre Côté, Tushar Khot, Erin Bransom, Bhavana Dalvi Mishra, Bodhisattwa Prasad Majumder, Oyvind Tafjord, and Peter Clark. DiscoveryWorld: A virtual environment for developing and evaluating automated scientific discovery agents. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024.

Peter Jansen, Oyvind Tafjord, Marissa Radensky, Pao Siangliulue, Tom Hope, Bhavana Dalvi Mishra, Bodhisattwa Prasad Majumder, Daniel S. Weld, and Peter Clark. CodeScientist: End-to-end semi-automated scientific discovery with code-based experimentation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 13370–13467, 2025.

Peter Jansen, Peter Clark, Doug Downey, and Daniel S. Weld. Generating literature-driven scientific theories at scale. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14677–14698, 2026.

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

Andrej Karpathy. CS231n lecture notes, Neural Networks Part 3: Babysitting the learning process. `https://cs231n.github.io/neural-networks-3/`, 2016.

Devvrit Khatri, Lovish Madaan, Rishabh Tiwari, Rachit Bansal, Venkata Sai Surya Subramanyam Duvvuri, Manzil Zaheer, Inderjit S. Dhillon, David Brandfonbrener, and Rishabh Agarwal. The art of scaling reinforcement learning compute for LLMs. In *International Conference on Learning Representations*, 2026.

Kimi Team. Kimi K3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tülu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Fenghai Li, Zihan Tang, Haofei Yu, Yining Zhao, and Jiaxuan You. Can large language models forecast what researchers study next? In *Conference on Empirical Methods in Natural Language Processing*, 2026.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Abbas, Cheng-Yu Hsieh, Dhruba Ghosh, Josh Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Chandu, Thao Nguyen, Igor Vasiljevic, Sham Kakade, Shuran Song, Sujay Sanghavi, Fartash Faghri, Sewoong Oh, Luke Zettlemoyer, Kyle Lo, Alaaeldin El-Nouby, Hadi Pouransari, Alexander Toshev, Stephanie Wang, Dirk Groeneveld, Luca Soldaini, Pang Wei Koh, Jenia Jitsev, Thomas Kollar, Alexandros G. Dimakis, Yair Carmon, Achal Dave, Ludwig Schmidt, and Vaishaal Shankar. DataComp-LM: In search of the next generation of training sets for language models. *arXiv preprint arXiv:2406.11794*, 2024.

Fanqi Lin, Yingdong Hu, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation. In *International Conference on Learning Representations*, 2025.

Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. Large language models to enhance Bayesian optimization. In *International Conference on Learning Representations*, 2024.

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

Kairong Luo, Haodong Wen, Shengding Hu, Zhenbo Sun, Zhiyuan Liu, Maosong Sun, Kaifeng Lyu, and Wenguang Chen. A multi-power law for loss curve prediction across learning rate schedules. In *International Conference on Learning Representations*, 2025a.

Xiaoliang Luo, Akilles Rechardt, Guangzhi Sun, Kevin K. Nejad, Felipe Yáñez, Bati Yilmaz, Kangjoo Lee, Alexandra O. Cohen, Valentina Borghesani, Anton Pashkov, Daniele Marinazzo, Jonathan Nicholas, Alessandro Salatiello, Ilia Sucholutsky, Pasquale Minervini, Sepehr Razavi, Roberta Rocca, Elkhan Yusifov, Tereza Okalova, Nianlong Gu, Martin Ferianc, Mikail Khona, Kaustubh R. Patil, Pui-Shee Lee, Rui Mata, Nicholas E. Myers, Jennifer K. Bizley, Sebastian Musslick, Isil Poyraz Bilgin, Guiomar Niso, Justin M. Ales, Michael Gaebler, N. Apurva Ratan Murty, Leyla Loued-Khenissi, Anna Behler, Chloe M. Hall, Jessica Dafflon, Sherry Dongqi Bao, and Bradley C. Love. Large language models surpass human experts in predicting neuroscience results. *Nature Human Behaviour*, 9(2):305–315, 2025b.

Marin Community. Marin: An open lab for building foundation models. https://github.com/marin-community/marin, 2025.

Meta Superintelligence Lab. Muse-Glimmer-30B. https://huggingface.co/meta-models/Muse-Glimmer-30B, 2026. Model card.

Igor Molybog, Peter Albert, Moya Chen, Zachary DeVito, David Esiobu, Naman Goyal, Punit Singh Koura, Sharan Narang, Andrew Poulton, Ruan Silva, Binh Tang, Diana Liskovich, Puxin Xu, Yuchen Zhang, Melanie Kambadur, Stephen Roller, and Susan Zhang. A theory on Adam instability in large-scale machine learning. *arXiv preprint arXiv:2304.09871*, 2023.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. OLMoE: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*, 2024.

Allan H. Murphy. Skill scores based on the mean square error and their relationships to the correlation coefficient. *Monthly Weather Review*, 116(12):2417–2424, 1988.

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

OpenAI. GPT-5.6 system card. https://deploymentsafety.openai.com/gpt-5-6, 2026. Model card.

Jungsoo Park, Ethan Mendes, Gabriel Stanovsky, and Alan Ritter. Anticipatory evaluation of language models. *arXiv preprint arXiv:2509.20645*, 2025.

Qwen Team. Qwen3-Next-80B-A3B-Instruct. https://huggingface.co/Qwen/Qwen3-Next-80B-A3B-Instruct, 2025. Model card.

Qwen Team. Qwen3.5: Towards native multimodal agents. https://qwen.ai/blog?id=qwen3.5, 2026. Blog post.

Herilalaina Rakotoarison, Steven Adriaensen, Neeratyoy Mallik, Samir Garibov, Eddie Bergman, and Frank Hutter. In-context freeze-thaw Bayesian optimization for hyperparameter optimization. In *International Conference on Machine Learning*, 2024.

James Requeima, John Bronskill, Dami Choi, Richard E. Turner, and David Duvenaud. LLM processes: Numerical predictive distributions conditioned on natural language. In *Advances in Neural Information Processing Systems*, 2024.

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

Udari Madhushani Sehwag, Elaine Lau, Haniyeh Ehsani Oskouie, Shayan Shabihi, Erich Liang, Andrea Toledo, Guillermo Mangialardi, Sergio Fonrouge, Ed-Yeremai Hernandez Cardona, Paula Vergara, Utkarsh Tyagi, Chen Bo Calvin Zhang, Pavi Bhatter, Nicholas Johnson, Furong Huang, Ernesto Gabriel Hernandez Montoya, and Bing Liu. SciPredict: Can LLMs predict the outcomes of scientific experiments in natural sciences? In *International Conference on Machine Learning*, 2026.

Chenglei Si, Diyi Yang, and Tatsunori Hashimoto. Can LLMs generate novel research ideas? A large-scale human study with 100+ NLP researchers. In *International Conference on Learning Representations*, 2025.

Chenglei Si, Tatsunori Hashimoto, and Diyi Yang. The ideation-execution gap: Execution outcomes of LLM-generated versus human research ideas. In *International Conference on Learning Representations*, 2026a.

Chenglei Si, Zitong Yang, Yejin Choi, Emmanuel Candès, Diyi Yang, and Tatsunori Hashimoto. Towards execution-grounded automated AI research. *arXiv preprint arXiv:2601.14525*, 2026b.

Xingyou Song, Oscar Li, Chansoo Lee, Bangding Yang, Daiyi Peng, Sagi Perel, and Yutian Chen. OmniPred: Language models as universal regressors. *Transactions on Machine Learning Research*, 2024.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. PaperBench: Evaluating AI's ability to replicate AI research. In *International Conference on Machine Learning*, 2025.

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

Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. Spike no more: Stabilizing the pre-training of large language models. In *Conference on Language Modeling*, 2025.

Mingtian Tan, Mike A. Merrill, Vinayak Gupta, Tim Althoff, and Thomas Hartvigsen. Are language models actually useful for time series forecasting? In *Advances in Neural Information Processing Systems*, 2024.

Kutay Tire, Ege Onur Taga, Muhammed Emrullah Ildiz, and Samet Oymak. Retrieval augmented time series forecasting. In *International Conference on Artificial Intelligence and Statistics*, 2026.

Howe Tissue, Venus Wang, and Lu Wang. Scaling law with learning rate annealing. In *Advances in Neural Information Processing Systems*, 2025.

Rosni Vasu, Peter Jansen, Pao Siangliulue, Cristina Sarasua, Abraham Bernstein, Peter Clark, and Bhavana Dalvi Mishra. HARPA: A testability-driven, literature-grounded framework for research ideation. *arXiv preprint arXiv:2510.00620*, 2025.

Heng Wang, Pengcheng Jiang, Jiashuo Sun, Zhiyi Shi, Haofei Yu, Jiawei Han, and Heng Ji. Learning to predict future-aligned research proposals with language models. *arXiv preprint arXiv:2603.27146*, 2026.

Jiaxin Wen, Chenglei Si, Yueh-han Chen, He He, and Shi Feng. Predicting empirical AI research outcomes with language models. In *Advances in Neural Information Processing Systems*, 2025.

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

Andrew Robert Williams, Arjun Ashok, Étienne Marcotte, Valentina Zantedeschi, Jithendaraa Subramanian, Roland Riachi, James Requeima, Alexandre Lacoste, Irina Rish, Nicolas Chapados, and Alexandre Drouin. Context is key: A benchmark for forecasting with essential textual information. In *International Conference on Machine Learning*, pp. 66887–66944, 2025.

Martin Wistuba, Arlind Kadra, and Josif Grabocka. Supervising the multi-fidelity race of hyperparameter configurations. In *Advances in Neural Information Processing Systems*, 2022.

Mitchell Wortsman, Peter J. Liu, Lechao Xiao, Katie Everett, Alex Alemi, Ben Adlam, John D. Co-Reyes, Izzeddin Gur, Abhishek Kumar, Roman Novak, Jeffrey Pennington, Jascha Sohl-Dickstein, Kelvin Xu, Jaehoon Lee, Justin Gilmer, and Simon Kornblith. Small-scale proxies for large-scale transformer training instabilities. In *International Conference on Learning Representations*, 2024.

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

Cheng Yan, Felix Mohr, and Tom Viering. LCDB 1.1: A database illustrating learning curves are more ill-behaved than previously thought. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, 2025.

Huaqing Zhang, Kaiyue Wen, and Tengyu Ma. Configuration-to-performance scaling law with neural ansatz. *arXiv preprint arXiv:2602.10300*, 2026.

Yiming Zhang, Athul Paul Jacob, Vivian Lai, Daniel Fried, and Daphne Ippolito. Human-aligned chess with a bit of search. In *International Conference on Learning Representations*, 2025.

AI USE STATEMENT

The project was conceived by the human authors. The data collection was designed by the authors and implemented with coding agents; the experiments were designed by the authors, with code partly written and jobs monitored by coding agents. The text was written by the authors and edited with AI assistance; every number and claim was verified by the authors.

ETHICS STATEMENT

All runs were collected from public sources, namely public experiment-tracker projects, public repositories on the Hugging Face Hub, and a public speedrun log archive, under the terms on which they were made public. Provenance (source and run identifier) is recorded per run; licenses were not audited run by run, and a license is recorded only for the speedrun archive (Appendix B). In the RL set, runs whose owners could not be tied to an institution with an associated paper or official repository were treated as requiring the authors' consent to redistribution and were excluded before any model was evaluated; the other sets were not screened for consent. Community runs are analyzed in aggregate and are not attributed to individual authors in the paper. The models' outputs are forecasts of training metrics and carry no personal or sensitive content.

REPRODUCIBILITY STATEMENT

Every number in the paper can be regenerated from the artifacts that will be released upon acceptance. The four test sets (150 pretraining, 150 fine-tuning, 150 RL, and 232 community runs) are frozen with content hashes before any model was evaluated, and Appendix B gives the sources, the filtering funnel, and the lineage isolation that separates test runs from every training pool; Appendix J.1 shows one complete question as the forecaster sees it, with its answer, and Appendix J the exact prompts. Appendix A.1 lists the identifier sent to each provider, the pinned commit of every locally served model and the provider identifier of every API-served model, including the three open-weight models served through a provider, the decoding settings, and the compute used; Appendix A.2 gives the implementation, checkpoint, and input contract of every baseline, and Appendix A.3 the rule that decides whether an answer is scored. The skill score is defined in §3.1 and worked through on an example in Appendix C; the paired contrasts behind the semantics, agent, reader-training, and database claims are tabulated with their resampling units in Appendix C.2, and the observed-share, input-window, failure-monitoring, matching, and community-set results are tabulated per method in Appendices F, E.2, and D.7; the reader is compared with the LLMs by unpaired per-stage scores (Appendix G.3). The reader's training recipe, pools, untrained baselines, three-stage averages, and ablations are in Appendix G, and the protocols of the agent, observed-share and failure-monitoring, and database analyses in Appendices I, F, and H. The benchmark, the scorer, the request and answer records of every API and open-weight model, and the trained reader adapters, each packaged with the hashes above, will be released upon acceptance.

A IMPLEMENTATION DETAILS

This appendix gives what is needed to reproduce every number in the paper: the model identifiers, decoding settings, and compute, the implementation of each baseline of §3.3, and the rules that decide whether an answer is valid. Every pretraining and fine-tuning question is asked on two query grids, GRID10 ($x = 0.3, 0.4, \ldots, 1.0$, the grid of §3.1 and the default) and GRID20 ($x = 0.4, 0.6, 0.8, 1.0$); RL scores pool both grids. Scores are aggregated over cells, one per stage and source: Marin, MLF, and AI2 pretraining, and Marin and HF fine-tuning (MLF = DCLM from MLFoundations, AI2 = OLMoE, HF = Open-R1; Table 1).

A.1 MODELS AND INFERENCE SETTINGS

All API models are called through the Bedrock Converse interface or the Gemini generative-language endpoint with a single sample per request and an output limit of 4,096 tokens. They use provider-default sampling (no temperature, nucleus, stop, or seed parameter), except that the two Gemini models run with thinking set to low; intervals are conditional on the one answer per request. The three open-weight models served through Bedrock (Kimi K3, Qwen3-Next-80B, and DeepSeek-V3.2)

Table 2: The 14 LLMs of the paper (Table 10), with the identifier actually sent to the provider. Every model answered the same frozen requests (the CURVE and FULL requests were frozen on 16 September 2026, the CURVE+CONFIG requests on 18 September). Locally served models are pinned to a Hugging Face commit, whose first eight characters are given; the models served through Bedrock or the Gemini API, including the open-weight Kimi K3, Qwen3-Next-80B, and DeepSeek-V3.2, are identified by the provider's model identifier only.

Short name	Identifier	Provider	Decoding
GPT-5.6 Sol	`us.openai.gpt-5.6-sol`	Bedrock	provider default
GPT-5.6 Luna	`us.openai.gpt-5.6-luna`	Bedrock	provider default
Claude Opus 5	`global.anthropic.claude-opus-5`	Bedrock	provider default
Claude Haiku 4.5	`global.anthropic.claude-haiku-4-5-20251001-v1:0`	Bedrock	provider default
Kimi K3	`global.moonshotai.kimi-k3`	Bedrock	single sample
Qwen3-Next-80B	`qwen.qwen3-next-80b-a3b`	Bedrock	provider default
DeepSeek-V3.2	`deepseek.v3.2`	Bedrock	provider default
Gemini 3.1 Pro	`gemini-3.1-pro-preview`	Gemini API	thinking low
Gemini 3.8 Flash	`gemini-3.8-flash`	Gemini API	thinking low
Muse-Glimmer-30B	`meta-models/Muse-Glimmer-30B` (`a4e59da5`)	local, HF	greedy, reasoning low
Qwen3.5-0.8B	`Qwen/Qwen3.5-0.8B` (`2fc06364`)	local, vLLM	greedy, no thinking
Qwen3.5-2B	`Qwen/Qwen3.5-2B` (`15852e8c`)	local, vLLM	greedy, no thinking
Qwen3.5-4B	`Qwen/Qwen3.5-4B` (`851bf6e8`)	local, HF	greedy, no thinking
Qwen3.5-9B	`Qwen/Qwen3.5-9B` (`c2022362`)	local, HF	greedy, no thinking

Table 3: What each non-LLM method sees and where it comes from. "Prefix" is the ten observed points; "pool" is our training runs, with the test run's sweep held out unless stated otherwise (Appendix D.8).

Method	Code and weights	Input	Output	Trained on
Last value	ours	last prefix point	point	–
Power-law fit	ours (SciPy)	prefix	point	–
AIC ensemble	ours (SciPy)	prefix	point	–
LC-PFN	authors' code and checkpoint	prefix	quantiles	synthetic prior (authors)
FT-PFN (ifBO)	authors' code and checkpoint	prefix	quantiles	synthetic prior (authors)
Chronos-2	released model	prefix, upsampled	quantiles	authors' pretraining
TimesFM-2.5	released model	prefix	quantiles	authors' pretraining
NeuNeu-NoLoss	authors' architecture, our training	prefix	quantiles	89 external runs
NCPL (released)	authors' checkpoint	eight config fields	point	authors' data
NCPL (retrained)	authors' code	config + prefix + budget	point	our pool
kNN on prefix shape, mean training shape	ours	prefix	point	our pool

are called in the same way. The five locally served models are decoded greedily with bf16 weights and a fixed seed; thinking is disabled for the four Qwen3.5 models, and Muse-Glimmer-30B runs with reasoning set to low. The 0.8B and 2B models were served by vLLM 0.29.0 and the 4B, 9B, and 30B models by Hugging Face Transformers 5.17.0. Run on the same 240 requests under both backends, Qwen3.5-9B gives identical parsed answers on 203 of the 227 that both parse, and its median forecasts differ by 0.021 on average (55 of 65 forecasting items identical).

Compute and release. Every local job in the paper runs on a single 80 GB GPU (A100 or H100) on academic clusters; no run uses more than one GPU, and the reader recipe trains in about 1.2 GPU-hours (Appendix G). The frozen requests, every raw answer, the scorer, and the trained reader adapters (LoRA weights on top of the pinned Qwen3.5 revisions) are packaged with their hashes and will be released with the benchmark upon acceptance.

A.2 BASELINES AND PRIOR METHODS: IMPLEMENTATION

Every baseline sees the same prefix as the LLMs: on pretraining and fine-tuning runs, the ten window means $\bar{y}_{1:10}$ at 2%, 4%, …, 20% of the planned budget; on RL runs, the dense prefix as released, from which the parametric fits take the ten 2% window means. The RL prefix comes in two forms: for 96 of the 150 runs it gives trailing 2% window means every 0.25% of the budget, and for the other 54 the raw logged points. The persistence forecast (the last value) repeats the last point of the prefix, which is $\bar{y}_{10}$ on pretraining and fine-tuning runs and on the 96 RL runs, and the raw last logged point on the other 54. Methods that output quantiles are scored on their median; the others give a point forecast only. Unless stated otherwise the method runs on CPU with a fixed seed, and every checkpoint is pinned by its hash. Table 3 summarizes what each of the 12 methods without an LLM in Table 11 sees.

Parametric fits. The power law is $a + b\,x^{-c}$ with $b \geq 0$ and $c \in [0.01, 5]$, fitted by bounded least squares from four starting exponents, keeping the lowest squared error. The AIC ensemble fits a power law, an exponential $a + b\,e^{-cx}$, and a log-linear law $a + b \ln x$, and averages their forecasts with weights $\exp(-\frac{1}{2}\Delta\mathrm{AIC})$. When every fit fails the forecast falls back to the last value. On RL runs the fits are applied to the negated reward, so that all curves are fitted as decreasing.

LC-PFN. We use the authors' code (automl/lcpfn at commit `ba892f6`) and their released six-layer checkpoint (`pfn_EPOCH1000_EMSIZE512_NLAYERS6_NBUCKETS1000`). The network was trained on synthetic learning curves indexed by epoch from 1 to 100, so we pass the fraction of the budget multiplied by 100 as the epoch. Values are mapped into the prior's range with the authors' `pfn_normalize`, using the minimum and maximum of the observed prefix as soft bounds and treating the curve as one to be minimized; RL rewards are negated first and the quantiles flipped back. Two small changes were needed to load the checkpoint in a current PyTorch release: the encoder loop passes the explicit attention mask as the original code did, and the GELU layers use the exact form they were trained with. The prior is saturating curves over epochs rather than LLM loss over tokens, and the soft bounds come from the prefix.

FT-PFN. We run the released ifBO checkpoint (`bopfn_broken_unisep_1000curves_10params_2M`) with a single curve in context, the run's own prefix, with time as the raw budget fraction. Values are mapped to $(0, 1)$ by a sigmoid centered on the prefix median and scaled by a robust spread, and mapped back through the logit. The model also takes ten hyperparameter slots; we fill the ones that match a fixed list of names and set the rest to the neutral value 0.5, so in practice it forecasts from the curve. Its 80% intervals are too narrow on our runs.

Time-series foundation models. Chronos-2 (`amazon/chronos-2`) reads its input in patches of 16 points, so ten points would fill less than one patch; we therefore interpolate the prefix linearly onto a grid of 0.2% of the budget (91 points) and forecast 400 steps to the end of the run. TimesFM-2.5 (`google/timesfm-2.5-200m`) receives the ten raw points and forecasts 40 steps of 2%; its context therefore sits inside a single input patch, a limitation we did not remove. Both models are used zero-shot, and their 10th, 50th, and 90th percentile outputs are taken as the forecast. On RL runs both receive the dense prefix interpolated onto 100 points.

NeuNeu-NoLoss. No trained checkpoint of this variant is released, so we trained the authors' architecture (six layers, width 512, about 19M parameters, five quantile heads) ourselves: 1,000 updates on 89 pretraining runs of an external source (billzid/predictable-scale), never on our runs. The model reads only the history of the loss; it was trained with a standardization fitted on the external source and tested with each prefix standardized by its own mean and standard deviation. The original paper predicts downstream accuracy, whereas we train it to predict the loss.

NCPL. The released checkpoint (`OptimizerStudy/NCPL-intermediate`) maps eight numeric configuration fields (model size, data size, learning rate, warmup, weight decay, batch size, number of steps, and the query fraction) to a residual on top of a Chinchilla-style loss offset; it has no way to mark a field as missing, so it can score only the 16 test runs that report all eight (Table 14). The retrained NCPL uses the authors' code and a Qwen3-1.7B backbone, and is adapted to our task: its input is the signed-log numeric configuration, the ten prefix points standardized by their own mean and standard deviation, and the budget fraction of the query, and a scalar head is trained with squared error on the standardized target. It trains for 100 updates of the head alone and 400 updates of the full model (AdamW, learning rates 5×10^{-5} and 10^{-5}), with sampling weights that balance sources, then runs, then query points, and no validation-based checkpoint selection. It is retrained once per regime; in the sweep-held-out regime every test sweep is removed from the pool at once, and in the source-held-out regime the test cell's whole source is removed, one model per cell. The community-run row and the ablations of §4.2 use one sweep-held-out model trained under this recipe, pool, and seed; on the ablations it is used as a fixed forecaster (the observed prefix is resampled to its ten input slots for shorter shares, its input is the 2% window by construction, and as a failure monitor it is scored by the predeclared adverse-change rule used for LC-PFN), never retrained per arm.

Pool-trained regressors. The two shape regressors predict each future value divided by the last observed value, as our reader does. The nearest-neighbor regressor (kNN on prefix shape) averages

Table 4: Answers accepted per LLM, out of 1,200 forecasting requests (300 runs, two grids, CURVE and FULL) and 2,400 matching requests (the 300 pairs of the frozen pretraining and fine-tuning release, two orders, two conditions, two grids). CURVE+CONFIG answers are not included (their coverage is in Appendix D.1). Strict requires the exact output schema; lenient requires only that the numbers parse and that crossed quantiles can be sorted; answers that pass lenient parsing but are absurd (any median outside the observed prefix's range widened by five ranges on each side; see text) are removed afterwards, before scoring, and are counted in the column "absurd" (a subset of lenient).

	Forecasting			Matching	
Model	strict	lenient	absurd	strict	lenient
Gemini 3.1 Pro	1198	1198	1	2392	2393
GPT-5.6 Luna	1196	1196	0	2390	2390
GPT-5.6 Sol	1192	1192	1	2398	2398
Muse-Glimmer-30B	1183	1198	4	2254	2399
Claude Haiku 4.5	1150	1198	8	2395	2395
Kimi K3	1139	1145	1	2386	2396
Gemini 3.8 Flash	982	984	5	2372	2391
Qwen3-Next-80B	880	1200	0	2398	2400
DeepSeek-V3.2	852	1163	19	2390	2392
Claude Opus 5	822	1181	0	1993	2373
Qwen3.5-0.8B	341	610	0	1561	1930
Qwen3.5-2B	203	572	0	1077	1576
Qwen3.5-9B	166	1199	1	2145	2208
Qwen3.5-4B	89	242	0	2040	2118

the future ratios of the five training runs whose standardized prefixes are closest, and the mean training shape applies the average future ratio of the pool. On pretraining and fine-tuning runs these two use the training split of the frozen release, with the test run's sweep removed per test run; on RL runs they use our training pool, with the target written as a change from the last value scaled by the prefix range, and the held-out variant removes the test run's whole project.

A.3 VALIDITY

Every error ratio in the paper is computed over the views a model answered validly (lenient); invalid answers are dropped rather than penalized. Per-cell counts are the superscripts of Table 10; Qwen3.5-4B has no GRID10 FULL score because it answers no full-context view in the AI2 pretraining cell.

Absurd outputs. After lenient parsing, a pretraining or fine-tuning forecast is removed as absurd when the median at any query point lies outside $[m - Kr,\ M + Kr]$ with $K = 5$, where m and M are the smallest and largest of the observed prefix values and $r = M - m$; a constant prefix ($r = 0$) is exempt, and none of the 600 frozen views has one. The rule reads only the prefix and the forecast, never the target, and removes the whole answer. It applies to the answers of every LLM, in every condition and grid, on the frozen pretraining and fine-tuning release (Tables 10 and 12), and to the reader, whose scorer uses the same rule; the methods without an LLM, and the RL, community-run, and failure-monitoring sets, are scored without it. It removes 40 of the 14,278 lenient forecasts in Table 4, 19 of them DeepSeek-V3.2's (1.6% of its forecasts), 8 Claude Haiku 4.5's, and at most 5 of any other model's (none for Claude Opus 5 and GPT-5.6 Luna); in the CURVE+CONFIG condition it removes 3 answers of Claude Haiku 4.5, 8 of DeepSeek-V3.2, 1 of Muse-Glimmer-30B, and none of Claude Opus 5, GPT-5.6 Sol, GPT-5.6 Luna, Gemini 3.1 Pro, Gemini 3.8 Flash, or Qwen3-Next-80B. The robustness check of Appendix D.3, which counts every invalid or missing answer as the last value ($R = 1$), treats a removed absurd forecast the same way, so the scores reported there include these answers (for every model but Kimi K3, which has no paired contrast and one removed answer), at the error of the last value; this is lenient to the model, because an absurd forecast's own error is usually larger. For the two models whose unfiltered scores were also computed, scoring the removed answers at their own error instead lowers S by at most 0.007 (Claude Haiku 4.5, GRID10 CURVE; 0.002 for GPT-5.6 Sol, GRID20 CURVE).

Table 4 motivates lenient validity: strict validity measures obedience to an output schema and not forecasting. Of the answers that the strict schema rejected and the lenient one accepted, 735 of Claude Opus 5's differ from a valid answer only in a missing or malformed rationale field, and 318 of Qwen3-Next-80B's are quantiles written in the wrong order. Scoring only the questions that every model answers under the strict schema makes the denominator both small and unrepresentative: on GRID20 it leaves 2 of the 25 AI2 pretraining questions, 10 of 25 on MLFoundations, and 946 of 1,800 information-condition pairs overall. Lenient validity does not rescue every model: the untrained Qwen3.5-0.8B, 2B, and 4B still fail to produce a parsable forecast on 49%, 52%, and 80% of the forecasting requests, and we report those rates rather than dropping the models. A valid answer can still be poor: of DeepSeek-V3.2's 287 curve-only pretraining answers, 81 invert the quantiles and 60 extrapolate the median far below every value in the prefix (0 and 7 for GPT-5.6 Sol), and its mean MAE there is 0.759 against 0.272 for GPT-5.6 Sol.

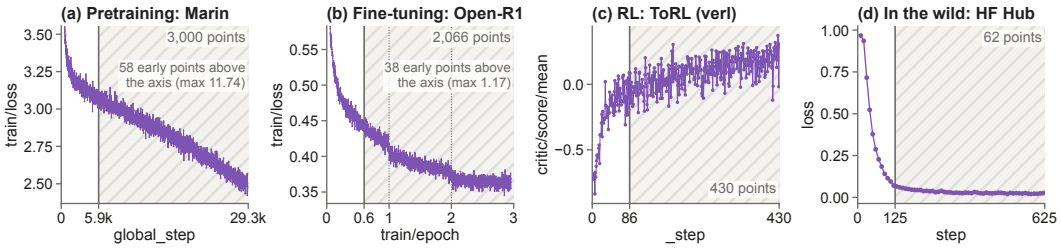

Figure 7: One test run per set, as logged. Every point inside the planned budget is drawn; the vertical line marks 20% of the budget, which the forecaster sees, and the shaded 80% is what it predicts. (a) Marin pretraining, 29,329 steps (30.8B tokens); our collector's sampled-history request returned 3,000 of its per-step records. (b) Open-R1 fine-tuning of a 7B checkpoint for three epochs; the loss drops at each epoch boundary (dotted). (c) GRPO on math from a 1.5B checkpoint; the mean verifier score is noisy and rising. (d) A community LoRA fine-tuning run whose trainer state logs 62 points. In (a) and (b) the y axis starts after the first 2% of the budget, and the panel gives the number of earlier points above it.

B Data collection and provenance

This appendix describes where the runs of §3.2 come from and how the pools were built: the sources, the funnel from collected runs to the training pool, and what the lineage isolation does and does not cover.

Where the runs come from. We enumerated training runs from three kinds of source. Public experiment-tracker projects contributed 19,650 run identities, counting metadata-only and failed lookups. On the Hugging Face Hub, a capped listing returned 47,929 distinct models; 14,370 carried a trainer-state file after quantized, converted, and bulk-uploaded repositories were removed, and 4,503 were fetched and parsed. Public log archives contributed 1,516 run logs from one speedrun project. Of these, 17,805 runs have usable numeric curves, and the funnel below starts from them.

What a run of each stage looks like. Figure 7 shows one test run per set as it was logged, with the 20% prefix that a forecaster sees and the 80% it must predict. The four stages differ in what the prefix reveals. A pretraining loss (a) falls steeply in the first few percent of the budget and then declines slowly and almost linearly for the rest; the prefix fixes the level, and the forecast turns on how fast the slow decline continues, which the learning-rate schedule and the token budget decide. A fine-tuning loss (b) steps down at every epoch boundary as the model sees the data again; the prefix ends inside the first epoch, so the size of those steps has to be read from the number of epochs and the size of the checkpoint rather than from the curve. An RL reward (c) rises through noise whose amplitude is as large as the trend over the prefix; the forecast is a statement about where the reward saturates, which the algorithm, the reward, and the budget govern. A community run (d) logs sparsely, here 62 points over 625 steps, and most of the curve is flat after the prefix, so repeating the last value is already a strong forecast and the skill of every method is low (§4.1). The logging density behind these panels is typical of each set: the median pretraining and fine-tuning run reaches the 3,000-row cap of our collector's sampled-history request (83 of 150 and 122 of 150 runs), the median RL run logs 518 points (interquartile range 200–1,953), and the median community run 308 (133–1,003).

From runs to a training pool. Table 5 is the funnel from the 17,805 runs to the pool the reader is trained on. A *sweep family* groups runs that differ only in the values a sweep varies (learning rate, seed, width, budget); its construction is given in Appendix D.8. Removing every run that shares a sweep family with any test run costs 1,654 runs, 82% of the 2,017 runs in the NCPL training and validation split that an NCPL-style predictor would otherwise train on; this is the step that matters for Appendix D.8.

Lineage and exposure. Splits isolate whole recorded *lineage components*, where a component joins runs connected by continuation, restart, or branch relations; intervals on pretraining and fine-tuning runs resample these components. Author identity alone does not form a component, so two unrelated runs by the same person may fall on opposite sides of the split; splits therefore do not guarantee author-level isolation. Validation targets 20% of the non-test pool, rounded to whole components.

Table 5: Training-pool funnel. Of the 3,024 clean runs, 2,055 can serve as forecast targets and 969 only as matching endpoints. The relaxed pool (1,793 runs) removes sweep families of test runs in one source only; the strict pool that supplies the reference reader recipe removes them in every source (1,463 runs, of which 1,105 can serve as forecast targets). The reader's 918 training runs are a subset of the strict pool's targets plus 340 public Hugging Face fine-tuning and RL runs admitted under the same rules (Appendix G).

Step	Rule	count
Universe	locally held runs with numeric curves	17,805
Eligible	the source's protocol can be applied	6,780
Test isolation	drop test run keys and test lineage components	6,552
Family isolation (one source)	drop the sweep family of any test run	4,898
Clean	curve-quality thresholds	3,024
Deduplicated	history hash; at most two seed replicates	1,835
Diverse (relaxed pool)	at most 8 per family, 25% per project	1,793
Strict pool	family isolation applied in every source	1,463

Selection uses identities and configurations, never forecasting performance, and no model was called before the sets were frozen. The frozen pretraining and fine-tuning test release contains 300 forecasting runs and 300 matching pairs (150 per stage) drawn from 589 distinct identities, whose lineage closure reserves 609; there are no cross-split violations. Its matching pairs are the ones counted in the validity table (Table 4, 2,400 matching requests per model). Its exclusion sets, used to build the training pool, are 1,089 test run keys and 649 test lineage components. Matching results (Appendix E.2) use a separate matching release that allows at most one reuse of any curve endpoint within a set. The wild set of 232 runs was drawn from 1,513 candidates after removing 32 with an author collision and 4 with a token collision against the other sets.

Licensing and consent. Every run in the evaluation sets is recorded with its provenance, the source (experiment-tracker project, Hugging Face repository, or log archive) and the run identifier, and the released files are pinned by content hashes. A license is recorded per source only for the speedrun log archive (MIT); experiment-tracker and Hugging Face runs carry no license field, so they are used under the terms on which their owners made them public, and their licenses were not audited run by run. Consent is screened only in the RL set. Before any model result was seen, every run and matching pair was assigned an institution tier: tier 1 for institutional accounts and for the official logs of a published paper or framework, tier 2 for named researchers with a verifiable affiliation; runs from individual or unattributable accounts, and tier-2 runs without an associated paper or official repository, were flagged as requiring the authors' consent to redistribution and excluded. The screen removed 5 run identities and 26 matching pairs from an earlier candidate set, and the freed slots were refilled by the same seeded draw; the 150 RL runs (138 tier 1, 12 tier 2) and the 120 matching pairs of the 96-run part (78 and 42) carry no consent flag. The pretraining and fine-tuning release and the community set were not screened for consent.

What the isolation does not cover. The exposure audit is over declared historical files and recorded relations: for the pretraining and fine-tuning prediction release it scanned 963 development-input files against the release's 3,729-identity graph by exact key and found no match; every record is marked as "no exact match in declared scope", not as certified unexposed, and the matching, RL, and community releases were not scanned. The test set is restricted to runs logged densely enough to support a ten-point prefix, which biases it toward projects that log often.

Table 6: Agreement between the ranking of 26 methods under MAE and their ranking under five other metrics (12–13 quantile-emitting methods for weighted quantile loss), as Kendall's τ over the ten combinations of grid and cell.

Metric against MAE	mean τ	min τ	cell at the minimum
RMSE	0.968	0.945	GRID10 FT/HF
Error ratio against the last value	0.898	0.834	GRID10 FT/HF
Final-point absolute error	0.891	0.822	GRID10 PT/AI2
Final-point relative error	0.770	0.237	GRID20 FT/HF
Weighted quantile loss	0.678	0.256	GRID20 FT/HF

C METRIC DETAILS AND STATISTICAL SIGNIFICANCE

This appendix shows how the skill score of §3.1 is computed on an example, how its edge cases are handled, and the paired significance tables behind the claims of §4.

C.1 METRIC STUDY

How the skill score is computed, on an example. Take one cell with two runs. On run A the forecast has an MAE of 0.05 over the query points and repeating the last value has 0.20, a ratio of 0.25; on run B the two MAEs are 0.30 and 0.20, a ratio of 1.5. The cell's error ratio is the geometric mean of the run ratios, $\sqrt{0.25 \times 1.5} = 0.61$, and the overall R is the geometric mean of the five cells' ratios, so each cell counts equally however many runs it holds; the skill score is $S = 1 - R$, here 0.39 for this cell. S is not the share of the summed error removed: the same two runs have a summed-error ratio of $(0.05 + 0.30)/(0.20 + 0.20) = 0.875$, which would read as 12.5% of the error removed. We use the geometric mean so that one run with a large loss scale cannot dominate a cell, and so that halving the error on one run and doubling it on another cancels out. The 95% intervals resample lineage components with replacement (2,000 resamples, fixed seed) and recompute the whole aggregation, and an interval $[L, U]$ on R is reported as $[1 - U, 1 - L]$ on S.

Zero and invalid denominators. The ratio is undefined when repeating the last value is exact, which happens only when the future of a run is constant. No pretraining or fine-tuning test run is affected. In 17 of the 150 RL runs the logged series never leaves its last observed value; these runs (reward in 14, training loss in 3) are excluded from the RL score for every method alike, never replaced by a small constant, so RL scores are over the remaining 133 runs; Table 15 lists which forecasts were exactly constant on them. Invalid answers are handled as in Appendix A.3; comparisons on common valid answers are in Appendix D.3.

Which metric we report could change who wins, so we ranked all 26 methods under six metrics in each of the ten cells (Table 6). RMSE, error ratio, and absolute final-point error rank methods essentially as MAE does, with mean τ between 0.89 and 0.97 and no cell below 0.82, although the top three methods are the same under all four metrics in only three of the ten cells. Relative final-point error and weighted quantile loss reorder methods on the Hugging Face fine-tuning cell ($\tau = 0.237$ and 0.256): several of its runs end near zero, which makes the relative error unstable, and weighted quantile loss is defined only on the 12 or 13 methods that emit quantiles.

C.2 STATISTICAL SIGNIFICANCE

Tables 7–9 report paired differences in S between two forecasters scored on the same questions, restricted to questions both answered validly; a positive ΔS favors the first. The 95% intervals are percentile bootstrap intervals from 2,000 resamples of lineage components (pretraining and fine-tuning), runs (RL), or owners (wild), and from 10,000 resamples for retrieval; they are not adjusted for multiplicity. Untrained-reader scores rest on the answers the untrained model produced validly. We report intervals, not p-values.

Table 7: Paired contrasts for the benchmark (§4.1): GRID10 for PT and FT, both grids for RL. ΔS is the first forecaster's skill minus the second's; n is the number of paired questions. Intervals are unadjusted 95% bootstrap intervals.

Contrast	Stage	ΔS	95% CI	n
*§4.1: strongest LLMs (*FULL*) minus LC-PFN*				
Claude Opus 5	PT	0.03	$[-0.03, 0.11]$	147
Claude Opus 5	FT	0.008	$[-0.099, 0.106]$	147
Claude Opus 5	RL	0.29	$[0.18, 0.41]$	251
GPT-5.6 Sol	PT	0.000	$[-0.066, 0.064]$	149
GPT-5.6 Sol	FT	0.10	$[0.05, 0.16]$	148
GPT-5.6 Sol	RL	0.21	$[0.09, 0.34]$	263
*§4.1: Claude Opus 5 minus GPT-5.6 Sol (*FULL*)*				
	PT	0.03	$[-0.02, 0.09]$	146
	FT	-0.09	$[-0.20, 0.00]$	145
	RL	0.08	$[0.01, 0.15]$	250
§4.1: semantics, FULL *minus* CURVE				
Claude Opus 5	PT	0.01	$[-0.04, 0.07]$	145
Claude Opus 5	FT	-0.19	$[-0.29, -0.10]$	143
Claude Opus 5	RL	0.08	$[0.02, 0.14]$	241
GPT-5.6 Sol	PT	0.03	$[-0.02, 0.08]$	149
GPT-5.6 Sol	FT	0.06	$[0.02, 0.11]$	148
GPT-5.6 Sol	RL	0.009	$[-0.048, 0.066]$	263
GPT-5.6 Luna	PT	0.11	$[0.05, 0.18]$	150
GPT-5.6 Luna	FT	-0.04	$[-0.08, 0.00]$	148
Gemini 3.1 Pro	PT	0.07	$[0.00, 0.12]$	149
Gemini 3.1 Pro	FT	-0.07	$[-0.12, -0.02]$	149
Gemini 3.8 Flash	PT	0.05	$[0.00, 0.11]$	110
Gemini 3.8 Flash	FT	-0.04	$[-0.11, 0.02]$	81
Claude Haiku 4.5	PT	0.17	$[0.09, 0.26]$	150
Claude Haiku 4.5	FT	0.12	$[0.07, 0.17]$	147
Muse-Glimmer-30B	PT	-0.13	$[-0.24, -0.03]$	149
Muse-Glimmer-30B	FT	-0.12	$[-0.19, -0.06]$	148

Table 8: Further paired contrasts, continuing Table 7. Untrained readers receive the same reader prompt; PT+FT pools the two stages. Intervals are unadjusted 95% bootstrap intervals, except in the trained-minus-untrained block, which gives the mean ΔS over three training seeds with the range over seeds in brackets; every per-seed 95% interval in that block lies above zero (lowest lower bound 0.19), and n is the per-seed number of paired questions.

Contrast	Stage	ΔS	95% CI	n
§4.1: semantics, FULL *minus* CURVE *(other open-weight models)*				
DeepSeek-V3.2	PT	-0.28	$[-0.56, 0.03]$	138
DeepSeek-V3.2	FT	-0.62	$[-0.88, -0.38]$	141
Qwen3-Next-80B	PT	0.03	$[-0.03, 0.10]$	150
Qwen3-Next-80B	FT	-0.02	$[-0.06, 0.01]$	150
Kimi K3	PT	-0.04	$[-0.11, 0.03]$	136
Kimi K3	FT	-0.07	$[-0.11, -0.02]$	131
§4.5: agent minus single call				
Claude Opus 5, local tools	PT	-0.01	$[-0.06, 0.03]$	95
Claude Opus 5, local tools	FT	0.09	$[0.04, 0.14]$	99
Claude Opus 5, local tools	RL	0.04	$[-0.11, 0.19]$	43
Claude Opus 5, local tools + search	PT	0.03	$[-0.02, 0.08]$	96
Claude Opus 5, local tools + search	FT	0.11	$[0.05, 0.17]$	98
Claude Opus 5, local tools + search	RL	0.07	$[-0.06, 0.20]$	43
GPT-5.6 Sol, local tools	PT	-0.03	$[-0.09, 0.03]$	80
GPT-5.6 Sol, local tools	FT	-0.06	$[-0.07, -0.02]$	83
GPT-5.6 Sol, local tools	RL	-0.09	$[-0.28, 0.10]$	42
GPT-5.6 Sol, local tools + search	PT	-0.04	$[-0.10, 0.02]$	81
GPT-5.6 Sol, local tools + search	FT	-0.004	$[-0.055, 0.060]$	86
GPT-5.6 Sol, local tools + search	RL	-0.09	$[-0.26, 0.08]$	43
§4.3: trained reader (8,727 runs) minus untrained; mean and range over three seeds				
Qwen3.5-2B, config	PT	12.81	$[12.80, 12.82]$	131
Qwen3.5-2B, config	FT	1.78	$[1.65, 1.86]$	148–149
Qwen3.5-2B, config	RL	3.20	$[3.05, 3.41]$	74
Qwen3.5-2B, config + semantics	PT	6.42	$[6.41, 6.44]$	44
Qwen3.5-2B, config + semantics	FT	0.53	$[0.35, 0.66]$	108–111
Qwen3.5-2B, config + semantics	RL	3.29	$[3.20, 3.41]$	77
Qwen3.5-9B, config	PT	20.24	$[20.20, 20.27]$	122
Qwen3.5-9B, config	FT	1.82	$[1.77, 1.86]$	150
Qwen3.5-9B, config	RL	2.46	$[2.12, 2.64]$	164
Qwen3.5-9B, config + semantics	PT	16.10	$[16.06, 16.14]$	110
Qwen3.5-9B, config + semantics	FT	1.23	$[1.20, 1.25]$	130–131
Qwen3.5-9B, config + semantics	RL	2.74	$[2.68, 2.84]$	162
§4.3: trained reader (8,727 runs, mean of 3 seeds), config + semantics minus config				
Qwen3.5-2B reader	PT+FT	0.002	$[-0.014, 0.017]$	296
Qwen3.5-2B reader	RL	-0.14	$[-0.28, -0.03]$	164
Qwen3.5-2B reader	Wild	-0.03	$[-0.07, 0.01]$	464
Qwen3.5-9B reader	PT+FT	0.02	$[0.01, 0.03]$	299
Qwen3.5-9B reader	RL	0.32	$[0.17, 0.50]$	164
Qwen3.5-9B reader	Wild	0.07	$[0.03, 0.11]$	464

Table 9: Further paired contrasts, continuing Table 8. Retrieval rows report the geometric mean of the ratio of errors with and without retrieval instead of ΔS. Intervals are unadjusted 95% bootstrap intervals.

Contrast	Stage	ΔS or ratio	95% CI	n
§4.4: retrieval over none, geometric-mean error ratio (< 1 *favors retrieval)*				
Claude Opus 5, 1,726 runs	PT+FT	0.58	$[0.47, 0.69]$	117
GPT-5.6 Sol, 17 runs (1%)	PT+FT	0.90	$[0.78, 1.04]$	122
GPT-5.6 Sol, 5% of library	PT+FT	0.91	$[0.77, 1.07]$	123
GPT-5.6 Sol, 25% of library	PT+FT	0.71	$[0.57, 0.88]$	123
GPT-5.6 Sol, 1,726 runs	PT+FT	0.61	$[0.47, 0.76]$	123
GPT-5.6 Sol, 1,726 runs, k=1	PT+FT	0.75	$[0.61, 0.93]$	123
GPT-5.6 Sol, 1,726 runs, k=5	PT+FT	0.63	$[0.49, 0.80]$	123
GPT-5.6 Sol, BM25	PT+FT	0.58	$[0.44, 0.74]$	123
GPT-5.6 Sol, prefix-shape kNN	PT+FT	0.62	$[0.48, 0.78]$	123
GPT-5.6 Sol, random runs	PT+FT	0.97	$[0.84, 1.13]$	123
GPT-5.6 Sol, BM25, own family allowed	PT+FT	0.19	$[0.13, 0.28]$	99
GPT-5.6 Luna, 1,726 runs	PT+FT	0.60	$[0.48, 0.75]$	124
GPT-5.6 Luna, BM25	PT+FT	0.54	$[0.41, 0.70]$	124
GPT-5.6 Luna, prefix-shape kNN	PT+FT	0.64	$[0.49, 0.82]$	123
GPT-5.6 Luna, random runs	PT+FT	0.77	$[0.64, 0.92]$	124
GPT-5.6 Luna, BM25, own family allowed	PT+FT	0.28	$[0.21, 0.39]$	123
*Wild: LLM (*FULL*) minus LC-PFN, core-211 cohort (Appendix D.7)*				
Claude Opus 5	Wild	0.24	$[0.07, 0.42]$	211
GPT-5.6 Sol	Wild	0.23	$[0.10, 0.39]$	211
GPT-5.6 Luna	Wild	0.21	$[0.09, 0.36]$	211

D MAIN RESULTS: FULL TABLES

This appendix gives the tables behind Figure 2 and §4.1: per-cell results, the comparison on common valid answers, the reinforcement-learning and community-run sets, and the trained predictors under the three held-out regimes.

D.1 MAIN TABLE

Tables 10 and 11 give the per-cell MAE and the skill score S of every method on the GRID10 query points, $x = 0.3, 0.4, \ldots, 1.0$; Table 10 holds the zero-shot LLMs under the three information conditions, and Table 11 the methods without an LLM and the predictors trained on past runs. MAE is in the loss units of each source and is read down a column; S is comparable across rows. Every entry of Tables 10, 11, and 12 is computed after the filter for absurd outputs, so the tables agree on every shared entry. The TEAREADER rows of Tables 11 and 12 are one training run of the reference recipe (Appendix G), which scores $S = 0.675$ on pretraining and fine-tuning and -0.311 on RL; Table 24 gives the mean over three training seeds on the same pool (0.659 and -0.332).

D.2 MAIN TABLE, GRID20

On the coarser GRID20, over the same runs and with the same scorer, the block order and the CURVE and FULL comparisons of §4.1 are unchanged, and the frontier models move by a few hundredths; Gemini 3.8 Flash ranks higher (error ratio 0.305 against 0.396) because its errors concentrate in the near half of the curve, which GRID20 does not query. The two GRID20 tables are in the release; RL entries do not depend on the grid.

D.3 COMMON VALID ANSWERS

Each model in Table 10 is scored on its own valid answers. Table 12 repeats the comparison on the 282 views that every model of the core group answered validly and gives, for every model, the paired difference to LC-PFN on the views both answered; no set of views is answered validly by every model, because the small open-weight models answer too few in the FULL condition. On the common views the ordering of Table 10 is unchanged, and only Claude Opus 5 with the curve alone is ahead of LC-PFN with an interval that excludes zero ($\Delta S = +0.08$). Counting every invalid, missing, or absurd answer (Appendix A.3) as the last value ($R = 1$ for that view) changes the scores of Claude Opus 5, GPT-5.6 Sol, GPT-5.6 Luna, Gemini 3.1 Pro, and Claude Haiku 4.5 by at most 0.01 and lowers the models with many invalid answers, such as Gemini 3.8 Flash (CURVE from 0.60 to 0.48, FULL from 0.60 to 0.56).

D.4 WHERE THE LLM BEATS THE CURVE FITTER

Table 13 asks on which runs the strongest LLMs gain over LC-PFN. The 294–297 GRID10 pretraining and fine-tuning views that both answered are split into quartiles of LC-PFN's error ratio R (its error over that of the last value), and within each quartile the table gives the geometric-mean R of both methods and the share of views on which the LLM is closer (per-view error ratios from the frozen scorer; geometric means over views, not over source cells as in Table 10). The gain is concentrated where the fitter fails: on the quartile where LC-PFN already cuts the last-value error by 70–98%, the LLMs are worse (1.35–1.71 times its error, closer on 38–43% of views), and on the other three quartiles they cut its error by 24–47% and are closer on 66–84%. Routing each view to whichever method errs less would raise the run-weighted skill from 0.69–0.71 (LLM alone) to 0.74–0.79, but the fitter's error is not observable before the answer, so a routing rule needs a signal computable from the prefix, which we leave to future work.

D.5 THE RELEASED NCPL CHECKPOINT

The released NCPL checkpoint needs eight numeric fields without a missing-value mask, which only 16 of the 300 test runs supply, all MLFoundations pretraining runs; Table 14 therefore scores every method on those 16 runs. The checkpoint sees the configuration only; it beats the last value, but its error ratio is about twice that of LC-PFN (0.48 against 0.23).

D.6 REINFORCEMENT LEARNING

The RL set holds 150 runs from more than 20 public projects, selected by a blind seeded rule (completion of at least 98% of the recovered budget, at most 25% of the set per project, at most two runs per group of identical configurations); its two parts of 96 and 54 runs share no run key and no project and differ in the form of the released prefix (Appendix A.2). RL scores exclude the 17 runs with a constant future (Appendix C; Table 15); the reader is scored on the 96-run part, 82 of whose runs remain. Reward units differ by project, so Table 16 gives the error per project in its own units beside the spread of the reward inside the observed prefix; its rows should not be pooled.

The two parts disagree on the methods without an LLM: on the 96 runs the AIC ensemble reaches an error ratio of 0.802 $[0.669, 0.971]$ and LC-PFN 0.806 $[0.684, 0.953]$, while the pool-trained shape regressors do not beat the last value (mean training shape 1.093 $[0.859, 1.405]$, nearest-neighbor regressor 1.247 $[1.021, 1.549]$, source held out); on the other 54 the shape regressors reach 0.74 and 0.80 and the AIC ensemble 1.262. Over all 133 scored runs, an exponential fit, which is not among the methods of Table 11, reaches $S = 0.14$ $[0.05, 0.23]$, against 0.13 $[0.01, 0.24]$ for LC-PFN. Gemini 3.1 Pro's low RL score comes from forecasts biased toward continued improvement: its median normalized error grows from $+0.06$ $[0.02, 0.11]$ at 30% of the budget to $+0.64$ $[0.34, 0.84]$ at the end, against at most $+0.03$ for Claude Opus 5 and GPT-5.6 Sol; its median path leaves the range the run ever occupied in 49% $[43, 55]$ of views (GPT-5.6 Sol 22%), and its 80% intervals cover 0.69 $[0.64, 0.74]$ of targets (GPT-5.6 Sol 0.85). This optimism helps only where the reward keeps rising: on the 30 PPO runs of the 54-run part, where the last value falls short by -1.04 $[-1.39, -0.87]$ at the end, Gemini 3.1 Pro is the only model not significantly short ($+0.25$ $[-0.09, 0.64]$). For curve fitters a longer prefix helps on RL (LC-PFN from an error ratio of 0.83 at 20% to 0.59 at 50% on 90 questions), and discarding the earliest segment, as Khatri et al. (2026) recommend, helps the parametric ensemble (1.37 to 0.80); neither helps the LLMs (GPT-5.6 Sol 0.68, 0.90, 0.67, 0.69 at 20–50%, and 0.73 without the early segment).

D.7 COMMUNITY RUNS IN THE WILD

The in-the-wild set holds 232 community runs by 184 authors, sampled from training-state files on the Hugging Face Hub (§3.2). Figure 2 scores the *core-211 cohort*: the 211 runs on which every model first evaluated on this set (GPT-5.6 Sol, GPT-5.6 Luna, and Claude Opus 5, each with the curve alone and with the full context, and the 2B TEAREADER) returned a valid GRID10 forecast; the other 21 runs lack a valid answer from at least one of them (Claude Opus 5 answered 222 of the 232 runs with the full context). The cohort was fixed before the other methods were run, and each method is scored on its own valid answers within it (199–211 runs per method). Scoring all 232 runs instead moves every skill score by at most 0.02 (Claude Opus 5 0.31, GPT-5.6 Sol 0.31, LC-PFN 0.09). By stage, the drop in skill concentrates in the 92 RL runs: over all 232 runs and both grids, the full-context error ratio of GPT-5.6 Sol, GPT-5.6 Luna, and Gemini 3.8 Flash is 0.96–1.04 on RL runs against 0.55–0.64 on the 140 fine-tuning and preference-tuning runs, and the hindsight-best smooth fit reaches an error ratio of 0.31 on the RL runs against 0.18 on the others. To test whether documentation explains the loss of skill, we edited the context of each run and forecast it again, paired within the run (GRID10). Deleting every identifying field (base model, datasets, reward functions, algorithm, framework) or adding everything recoverable offline from the model cards moves the error ratio by at most 4% for GPT-5.6 Sol, GPT-5.6 Luna, and Claude Opus 5 (every interval covers 1); removing the context altogether raises it by 14–16% for GPT-5.6 Sol and GPT-5.6 Luna (1.16 and 1.14) and leaves Claude Opus 5 at 0.99. Runs by individuals are forecast worse than runs by institutions (error ratios 0.81–1.02 against 0.44–0.64 for GPT-5.6 Sol, GPT-5.6 Luna, and Gemini 3.8 Flash in both conditions), and the gap is already present with the curve alone (1.6–1.8×); dividing each forecast by the error of the best smooth fit in hindsight for that run removes it (0.87–1.15, every interval covering one).

The logging explains the rest. The hindsight-best fit reaches an error ratio of 0.23 on community runs against 0.09 on the frozen sets. A prefix window rests on a median of 7 raw points in the wild against 59 in the frozen sets; thinning each frozen curve to the wild density raises its hindsight floor from 0.090 to 0.132, and adding wild-level residual noise raises it to 0.214 against the 0.225 observed, which reproduces 95% of the gap on a log scale (92% on a linear one). Regressing the floor on these curve properties shrinks the author term from $+0.49$ (t=2.9) to $+0.14$ (t=1.1). Creation date relative

to a model's knowledge cutoff has no detectable effect within stage ($|t| \leq 1.9$), but the test detects only a 27–52% change per year.

D.8 TRAINED PREDICTORS UNDER THREE REGIMES

A predictor trained on our pool is evaluated in one of three *held-out regimes*: with other runs of the test run's own sweep in the pool (*sweep seen*), with the test run's sweep family removed (*sweep held out*, the default), or with its whole source removed (*source held out*). For sources that name their runs, the sweep family is the run name with the trailing hash stripped and every numeric token replaced by a placeholder, so that seed, learning rate, width, budget, and date collapse into one family (`muonh-may-recipe-lr-v1-d768-R120-lr0p4` becomes `muonh-may-recipe-lr-v#-d#-R#-lr#p#`); for sources that expose only tracker identifiers, it is a signature of the configuration without the keys a sweep varies (learning rate, seed, weight decay, warmup, budget) and without infrastructure keys. A source is one tracker entity, so pretraining and fine-tuning runs of one organization are different sources. When a regime empties the pool, the method falls back to the last value and is scored normally. Table 17 gives the per-cell MAE.

With the sweep seen, the nearest-neighbor regressor reaches $S = 0.85$ and the mean training shape 0.73 (0.98 and 0.91 on the fine-tuning source whose 100 test runs come from one sweep), which is within-sweep interpolation: most of the pool shares a sweep family with a test run (82% of the NCPL training and validation split; Appendix B). Holding out the sweep family lowers them to 0.57 and 0.50, removing the wider sweep to 0.55 and 0.50, and holding out the source gives 0.38 and 0.58, all below LC-PFN (0.62); retrained NCPL falls from 0.79 with the test run's own sweep seen to 0.28 with its sweep family held out and 0.46 with the source held out. On Marin pretraining the better shape regressor reaches an MAE of 0.105 $[0.076, 0.141]$ with the sweep seen, 0.165 $[0.112, 0.227]$ with it held out, and 0.188 $[0.149, 0.230]$ with the source held out, against 0.150 $[0.106, 0.200]$ for the best zero-shot LLM. Three limits qualify the table. The strict pool contains no runs from the MLFoundations, AI2, or Hugging Face sources, so for those cells NCPL's sweep-held-out and source-held-out models were trained on the same rows and differ only by nondeterminism. The Marin fine-tuning test runs come from one sweep that holds 69 of the 71 pool runs for that cell, so with the sweep held out both shape regressors are fitted on the remaining two runs (0.214, against 0.097 for the last value). The source-held-out pool is 93–98% Marin pretraining for the non-Marin cells, and the best method is chosen per column after the fact, which flatters the trained block.

Table 10: Curve prediction on the frozen test release, GRID10: zero-shot LLMs (methods without an LLM in Table 11). MAE per cell in the loss units of each source (superscript: scored views when below the cell size 100/25/25/100/50) and skill score S with 95% intervals over lineage components. Sources: MLF = DCLM (MLFoundations), AI2 = OLMoE, HF = Open-R1 (Table 1); PT+FT pools the five pretraining and fine-tuning cells. Every entry excludes absurd outputs (Appendix A.3). Bold: best zero-shot entry in a column; underline: best across both halves. N/A: not run, or no valid answer in some cell. ‡598 and 599 of 600 requests answered.

	Pretraining MAE			Fine-tuning MAE			
Model, condition	Marin	MLF	AI2	Marin	HF	PT+FT $S \uparrow$	RL $S \uparrow$
Claude Opus 5							
CURVE	0.170^{98}	0.191	0.404	0.018^{98}	$\mathbf{0.057}^{47}$	**0.706** [0.67,0.74]	0.326 [0.19,0.44]
CURVE+CONFIG	0.158^{96}	0.190	0.290^{24}	0.020^{98}	0.071^{48}	0.704 [0.66,0.74]	N/A
FULL	$\mathbf{0.150}^{99}$	0.213^{24}	$\mathbf{0.216}^{24}$	0.022^{97}	0.119	0.648 [0.61,0.68]	**0.427** [0.33,0.51]
GPT-5.6 Sol							
CURVE	0.234	0.208	0.428	0.031	0.078	0.612 [0.57,0.66]	0.335 [0.23,0.43]
CURVE+CONFIG	0.222	0.190	0.447	0.029	0.074	0.623 [0.58,0.66]	N/A
FULL	0.193^{99}	0.228	0.411	0.028^{98}	0.069	0.657 [0.61,0.70]	0.346 [0.25,0.44]
GPT-5.6 Luna							
CURVE	0.264	0.202	0.457	0.032	0.083	0.574 [0.52,0.62]	0.180 [0.07,0.28]
CURVE+CONFIG	0.225	0.227	0.379	0.036	0.074	0.589 [0.54,0.63]	N/A
FULL	0.212	**0.159**	0.320	0.041^{99}	0.076^{49}	0.632 [0.59,0.67]	0.259 [0.16,0.35]
Gemini 3.1 Pro							
CURVE	0.249^{99}	0.316	0.441	0.020	0.074^{49}	0.606 [0.56,0.65]	0.012 [-0.17,0.16]
CURVE+CONFIG	0.219	0.324	0.430	0.017	0.068	0.614 [0.57,0.66]	N/A
FULL	0.198	0.278	0.350	0.032	0.062	0.610 [0.56,0.65]	-0.154 [-0.44,0.06]
Gemini 3.8 Flash							
CURVE	0.216^{82}	0.336^{17}	0.480^{18}	0.023^{65}	0.075^{35}	0.598 [0.54,0.65]	0.293 [0.18,0.39]
CURVE+CONFIG	0.175^{76}	0.290^{13}	0.471^{17}	$\mathbf{0.015}^{47}$	0.060^{26}	0.650 [0.60,0.69]	N/A
FULL	0.176^{92}	0.315^{23}	0.413^{23}	0.030^{84}	0.063^{43}	0.604 [0.55,0.65]	0.242 [0.11,0.35]
Claude Haiku 4.5							
CURVE	0.254	0.408	0.572	0.049	0.094^{47}	0.463 [0.40,0.52]	-0.045 [-0.19,0.09]
CURVE+CONFIG	0.185	0.382	0.525	0.041	0.073^{49}	0.550 [0.49,0.60]	N/A
FULL	0.229	0.248	0.428	0.032	0.078^{47}	0.608 [0.55,0.66]	0.057 [-0.06,0.17]
DeepSeek-V3.2							
CURVE	0.693^{97}	0.874^{21}	1.005^{23}	0.127^{98}	0.094^{47}	-0.035 [-0.19,0.10]	-0.007 [-0.15,0.11]
CURVE+CONFIG‡	0.806^{98}	0.980^{23}	0.943^{24}	0.126^{98}	0.105^{48}	-0.191 [-0.33,-0.05]	N/A
FULL	0.701^{99}	0.897^{24}	0.973^{23}	0.212^{98}	0.132^{46}	-0.467 [-0.64,-0.30]	-0.113 [-0.28,0.03]
Qwen3-Next-80B							
CURVE	0.283	0.230	0.366	0.047	0.075	0.540 [0.50,0.58]	-0.013 [-0.12,0.10]
CURVE+CONFIG‡	0.259	0.254	0.288	0.049	0.078^{49}	0.545 [0.51,0.58]	N/A
FULL	0.245	0.202	0.328	0.050	0.074	0.553 [0.52,0.58]	-0.037 [-0.15,0.08]
Muse-Glimmer-30B							
CURVE	0.221	0.431	0.407^{24}	0.027	0.064^{49}	0.580 [0.53,0.63]	0.125 [-0.02,0.25]
CURVE+CONFIG	0.246	0.439	0.489	0.032^{99}	0.064^{49}	0.539 [0.49,0.59]	N/A
FULL	0.264	0.447	0.478	0.050^{99}	0.073^{49}	0.446 [0.37,0.51]	0.102 [-0.03,0.21]
Qwen3.5-9B							
CURVE	0.368	0.460	0.566	0.062	0.070	0.379 [0.30,0.45]	-0.239 [-0.41,-0.09]
FULL	0.278^{99}	0.421	0.459	0.069	0.078	0.415 [0.34,0.48]	-0.155 [-0.28,-0.04]
Qwen3.5-2B							
CURVE	0.306^{87}	0.199^{24}	0.315	0.055^{95}	0.129^{42}	0.480 [0.43,0.53]	N/A
CURVE+CONFIG	0.277^{98}	0.207	0.340^{20}	0.062^{96}	0.136	0.454 [0.39,0.51]	N/A
FULL	0.665^{19}	0.388^{5}	0.247^{5}	0.048^{26}	0.281^{4}	0.306 [0.13,0.42]	N/A
Qwen3.5-4B							
CURVE	0.346^{49}	0.516^{10}	1.273^{2}	0.079^{16}	0.058^{21}	0.136 [0.01,0.25]	N/A
FULL	0.281^{17}	0.638^{4}	N/A	0.036^{6}	0.074^{8}	N/A	N/A
Qwen3.5-0.8B							
CURVE	0.797	0.840	0.838^{23}	0.104	0.223^{41}	-0.371 [-0.49,-0.27]	N/A
FULL	N/A	4.042^{15}	N/A	N/A	0.431^{1}	N/A	N/A
Kimi K3							
CURVE	0.249^{98}	0.202^{24}	0.303^{24}	0.029^{99}	0.075^{47}	0.658 [0.62,0.70]	0.264 [0.17,0.35]
FULL	0.200^{93}	0.267^{22}	0.323^{24}	0.037^{91}	0.069^{44}	0.608 [0.56,0.65]	0.229 [0.13,0.32]

Table 11: Curve prediction on the frozen test release, GRID10 ($x = 0.3, \ldots, 1.0$): methods without an LLM and predictors trained on past runs; columns as in Table 10. The last block has seen the sweep of the test run in training, which amounts to within-sweep interpolation (Appendix D.8), and is not part of the main comparison.

	Pretraining MAE			Fine-tuning MAE			
Method	Marin	MLF	AI2	Marin	HF	PT+FT $S \uparrow$	RL $S \uparrow$
Zero-shot, curve only							
Last value	0.533	0.495	0.575	0.097	0.108	0.000 [0.00,0.00]	0.000 [0.00,0.00]
Power-law fit	0.349	0.258	0.407	0.042	0.084	0.437 [0.38,0.49]	0.037 [-0.09,0.14]
AIC ensemble of fits	0.377	0.258	0.475	0.042	0.095	0.397 [0.34,0.45]	0.047 [-0.10,0.17]
LC-PFN	0.252	0.185	0.480	0.037	0.074	0.621 [0.56,0.67]	0.133 [0.01,0.24]
FT-PFN (ifBO)	0.368	0.416	0.483	0.071	0.100	0.286 [0.25,0.32]	-0.005 [-0.10,0.08]
Chronos-2	0.595	0.624	0.844	0.050	0.111	0.172 [0.09,0.25]	0.075 [-0.09,0.21]
TimesFM-2.5	0.839	0.682	1.120	0.138	0.136	-0.457 [-0.51,-0.41]	-0.385 [-0.62,-0.19]
Trained on past runs, test sweep held out							
TEAREADER (2B, ours)	0.123	0.247	0.275	0.031	<u>0.056</u>	0.675 [0.64,0.70]	-0.311 [-0.53,-0.13]†
NCPL (retrained)	0.216	0.384	0.513	0.061	0.191	0.279 [0.20,0.35]	-0.053 [-0.21,0.07]
NCPL (retrained, source held out)	0.303	0.328	0.435	0.032	0.094	0.460 [0.41,0.51]	N/A
NeuNeu-NoLoss (our training)	0.393	0.330	0.510	0.090	0.094	0.275 [0.24,0.31]	N/A
NCPL (released checkpoint)§	N/A	0.219[16]	N/A	N/A	N/A	N/A	N/A
kNN on prefix shape	0.165	0.205	0.195	0.214	0.108	0.570 [0.51,0.63]	-0.062 [-0.26,0.10]
Mean training shape	0.183	0.237	0.197	0.214	0.110	0.498 [0.42,0.56]	0.064 [-0.12,0.22]
Trained on past runs, test sweep seen							
NCPL (retrained)	0.192	<u>0.152</u>	0.234	0.005	0.082	0.792 [0.77,0.82]	N/A
kNN on prefix shape	<u>0.105</u>	0.172	<u>0.193</u>	<u>0.004</u>	0.110	<u>0.847</u> [0.82,0.87]	-0.062 [-0.26,0.10]
Mean training shape	0.182	0.231	0.195	0.009	0.113	0.734 [0.69,0.77]	0.062 [-0.12,0.21]

†The 96-run part of the RL set only; other RL entries use the whole set. NeuNeu-NoLoss is trained on 89 runs of an external source and never on ours (Appendix A.2). N/A: not evaluated. §The released NCPL checkpoint sees the configuration only and needs eight numeric fields that 16 of the 300 test runs supply (all MLFoundations pretraining); its one MAE is over those 16 runs and is not comparable with its column (same-run comparison in Table 14).

Table 12: Skill score on common valid answers, pretraining and fine-tuning, GRID10. Top: the 282 views that every model in the core group (the rows of the top block) answered validly. Bottom: each model on its own valid views, as in Table 10. ΔS is the paired difference to LC-PFN on the same views (positive: better than LC-PFN), with a 95% interval from a lineage-cluster bootstrap that resamples both methods together. Every entry excludes absurd outputs, as in Table 10. Kimi K3 CURVE (292 views, $S = 0.658$) and the CURVE+CONFIG rows of other models have no paired contrast in the final readout and appear only in Table 10.

Views	Method	n	S	ΔS vs. LC-PFN
Core group, common views				
	Claude Opus 5· CURVE	282	0.709	+0.083 [+0.04, +0.13]*
	TEAREADER (ours)	282	0.677	+0.051 [-0.00, +0.11]
	GPT-5.6 Sol· FULL	282	0.661	+0.036 [-0.01, +0.08]
	Claude Opus 5· FULL	282	0.646	+0.021 [-0.04, +0.08]
	GPT-5.6 Luna· FULL	282	0.638	+0.012 [-0.03, +0.07]
	GPT-5.6 Sol· CURVE+CONFIG	282	0.626	+0.001 [-0.04, +0.04]
	LC-PFN	282	0.626	–
	GPT-5.6 Sol· CURVE	282	0.611	-0.015 [-0.06, +0.03]
	GPT-5.6 Luna· CURVE+CONFIG	282	0.595	-0.031 [-0.09, +0.02]
	GPT-5.6 Luna· CURVE	282	0.573	-0.053 [-0.11, +0.00]
Each model, own valid views				
	kNN on prefix shape (sweep seen)	300	0.847	+0.226 [+0.17, +0.29]*
	NCPL, retrained (sweep seen)	300	0.792	+0.172 [+0.12, +0.22]*
	Mean training shape (sweep seen)	300	0.734	+0.113 [+0.05, +0.18]*
	Claude Opus 5· CURVE	293	0.706	+0.083 [+0.04, +0.13]*
	TEAREADER (ours)	299	0.675	+0.052 [+0.00, +0.11]*
	GPT-5.6 Sol· FULL	297	0.657	+0.037 [-0.01, +0.08]
	Claude Opus 5· FULL	294	0.648	+0.026 [-0.04, +0.09]
	GPT-5.6 Luna· FULL	298	0.632	+0.011 [-0.03, +0.06]
	GPT-5.6 Sol· CURVE+CONFIG	300	0.623	+0.003 [-0.04, +0.05]
	LC-PFN	300	0.621	–
	GPT-5.6 Sol· CURVE	300	0.612	-0.008 [-0.06, +0.04]
	Gemini 3.1 Pro· FULL	300	0.610	-0.011 [-0.07, +0.05]
	Kimi K3 · FULL	274	0.608	-0.015 [-0.07, +0.04]
	Claude Haiku 4.5 · FULL	297	0.608	-0.014 [-0.08, +0.04]
	Gemini 3.1 Pro· CURVE	298	0.606	-0.015 [-0.07, +0.04]
	Gemini 3.8 Flash· FULL	265	0.604	-0.004 [-0.07, +0.06]
	Gemini 3.8 Flash· CURVE	217	0.598	-0.045 [-0.10, +0.01]
	GPT-5.6 Luna· CURVE+CONFIG	300	0.589	-0.031 [-0.08, +0.02]
	Muse-Glimmer-30B · CURVE	298	0.580	-0.045 [-0.11, +0.02]
	GPT-5.6 Luna· CURVE	300	0.574	-0.047 [-0.10, +0.01]
	kNN on prefix shape (sweep held out)	300	0.570	-0.051 [-0.14, +0.03]
	Qwen3-Next-80B · FULL	300	0.553	-0.067 [-0.12, -0.01]*
	Qwen3-Next-80B · CURVE	300	0.540	-0.080 [-0.13, -0.03]*
	Mean training shape (sweep held out)	300	0.498	-0.122 [-0.22, -0.03]*
	Qwen3.5-2B · CURVE	273	0.480	-0.141 [-0.21, -0.06]*
	Claude Haiku 4.5 · CURVE	297	0.463	-0.159 [-0.23, -0.09]*
	NCPL, retrained (source held out)	300	0.460	-0.160 [-0.23, -0.10]*
	Qwen3.5-2B · CURVE+CONFIG	289	0.454	-0.172 [-0.26, -0.08]*
	Muse-Glimmer-30B · FULL	298	0.446	-0.175 [-0.26, -0.09]*
	Power-law fit	300	0.437	-0.183 [-0.24, -0.11]*
	Qwen3.5-9B · FULL	299	0.415	-0.205 [-0.28, -0.13]*
	AIC ensemble of fits	300	0.397	-0.224 [-0.29, -0.15]*
	Qwen3.5-9B · CURVE	300	0.379	-0.241 [-0.33, -0.15]*
	Qwen3.5-2B · FULL	59	0.306	-0.200 [-0.42, -0.05]*
	FT-PFN	300	0.286	-0.334 [-0.37, -0.29]*
	NCPL, retrained (sweep held out)	300	0.279	-0.342 [-0.44, -0.25]*
	NeuNeu-NoLoss	300	0.275	-0.346 [-0.39, -0.29]*
	Chronos-2	300	0.172	-0.449 [-0.52, -0.38]*
	Qwen3.5-4B · CURVE	98	0.136	-0.410 [-0.54, -0.28]*
	DeepSeek-V3.2 · CURVE	286	-0.035	-0.667 [-0.83, -0.53]*
	Qwen3.5-0.8B · CURVE	289	-0.371	-0.991 [-1.11, -0.88]*
	TimesFM-2.5	300	-0.457	-1.078 [-1.15, -1.00]*
	DeepSeek-V3.2 · FULL	290	-0.467	-1.098 [-1.28, -0.92]*

* interval excludes zero.

Table 13: LLM against LC-PFN by quartile of LC-PFN's error ratio R (GRID10, FULL prompt, pretraining and fine-tuning views both methods answered). Geometric means over views; "closer" is the share of views on which the LLM's error is smaller.

		Claude Opus 5 (n=294)			GPT-5.6 Sol (n=297)		
LC-PFN R quartile	LC-PFN R	LLM R	LLM / LC-PFN	closer	LLM R	LLM / LC-PFN	closer
Q1 (0.02–0.30)	0.15	0.26	1.71	38%	0.21	1.35	43%
Q2 (0.30–0.39)	0.34	0.22	0.64	75%	0.26	0.76	66%
Q3 (0.39–0.59)	0.47	0.25	0.53	78%	0.32	0.68	84%
Q4 (0.59–2.07)	0.84	0.50	0.59	68%	0.56	0.66	81%

Table 14: The released NCPL checkpoint on the 16 MLFoundations pretraining runs it can score; every row is scored on the same 16 runs. MAE is in that source's loss units. The error ratio R is the geometric mean of MAE relative to the last value. Intervals are 95% cluster bootstraps over the 10 lineage components. The released checkpoint sees only the configuration, and every other row also sees the first 20% of the curve. We report the released checkpoint with the better of its two loss offsets, and the other offset is the second row.

	GRID10		GRID20	
Method	MAE	$R \downarrow$	MAE	$R \downarrow$
NCPL (released checkpoint)	0.219 [0.13,0.33]	0.477 [0.34,0.68]	0.234 [0.12,0.39]	0.428 [0.29,0.67]
NCPL (released, StepLaw offset)	0.776 [0.61,1.01]	1.921 [1.35,2.93]	0.800 [0.64,1.03]	1.825 [1.28,2.78]
Last value	0.459 [0.36,0.55]	1.000	0.500 [0.38,0.60]	1.000
Power-law fit	0.280 [0.20,0.34]	0.642 [0.52,0.79]	0.317 [0.22,0.40]	0.647 [0.52,0.79]
LC-PFN	0.185 [0.08,0.34]	0.230 [0.16,0.34]	0.218 [0.07,0.39]	0.228 [0.16,0.35]
Claude Opus 5 · CURVE	0.149 [0.09,0.22]	0.258 [0.20,0.33]	0.188 [0.10,0.28]	0.242 [0.17,0.34]
GPT-5.6 Sol · FULL	0.155 [0.08,0.25]	0.228 [0.17,0.35]	0.198^{15} [0.11,0.28]	0.269 [0.21,0.37]
GPT-5.6 Luna · FULL	0.148 [0.08,0.21]	0.260 [0.16,0.35]	0.186 [0.10,0.27]	0.249 [0.19,0.32]
NCPL (retrained, sweep held out)	0.293 [0.18,0.42]	0.553 [0.31,0.84]	0.344 [0.20,0.50]	0.565 [0.30,0.90]
TEAREADER (ours)	0.184 [0.14,0.23]	0.420 [0.33,0.57]	0.179 [0.12,0.25]	0.342 [0.28,0.42]

Table 15: The 17 RL runs whose logged series (reward in 14, training loss in 3 open-r1 runs) never leaves its last observed value, where the error ratio does not exist. For four LLMs and the reader, per condition: how many of these runs it answered, and on how many it forecast the constant exactly (zero error). Per-source errors in the series' own units are in the released analysis files. *The reader is scored on the 96-run part of the RL set, which holds 14 of the 17.

Model	Condition	Runs answered	Exactly constant
Claude Haiku 4.5	CURVE	17	12
Claude Haiku 4.5	FULL	17	6
GPT-5.6 Luna	CURVE	17	15
GPT-5.6 Luna	FULL	17	9
Claude Opus 5	CURVE	17	4
Claude Opus 5	FULL	17	10
TEAREADER (ours)	CURVE+CONFIG	12*	9
GPT-5.6 Sol	CURVE	17	16
GPT-5.6 Sol	FULL	17	16

Table 16: Forecasting error per project on the 96-run part of the RL set, in the reward units of that project, over both grids. "Range" is the mean spread of the reward inside the observed prefix and sets the scale of the row.

Project	runs	range	Claude Opus 5 Curve	Claude Opus 5 Full	GPT-5.6 Sol Full	Gemini 3.1 Pro Full
ai2-llm/Olmo-3-7B-RL-Zero	3	4.28	1.107	1.126	1.108	1.938
ai2-llm/open_instruct_public	11	4.30	0.336	0.336	0.446	0.951
apiche/watermelon	13	0.16	0.104	0.106	0.086	0.151
axolotl-ai/gsm8k-grpo-r1	1	1.88	0.280	0.056	0.358	1.219
cjreinforce/openrlhf_train_ppo	3	0.20	0.139	0.116	0.117	0.105
costa-huang/huggingface	6	1.33	0.266	0.230	0.238	0.411
dongfu/acecoder	1	1.14	0.044	0.021	0.053	0.224
dongfu/torl	1	0.93	0.045	0.066	0.062	0.265
dzmitry-bahdanau/watermelon	3	0.25	0.036	0.022	0.019	0.029
ehsk/watermelon	2	0.49	0.020	0.021	0.021	0.108
hamishivi/tmax-public	1	0.00	0.077	0.001	0.000	0.088
hashimoto-group/DeepScholarRL-ablations	1	5.47	0.489	0.269	0.200	0.716
hashimoto-group/DeepScholarRLTTT-ablations	3	0.00	0.067	0.021	0.005	0.341
hashimoto-group/metagrad-rl	3	0.03	0.125	0.121	0.121	0.130
hashimoto-group/verl_examples	1	0.09	0.078	0.088	0.109	0.234
huggingface/trl	6	0.18	0.038	0.018	0.017	0.044
jiayipan/tinyrl	6	0.30	0.156	0.110	0.131	0.207
lkevinzc/gem	13	0.59	0.059	0.044	0.063	0.166
lkevinzc/gem-tinker_train	5	0.33	0.064	0.032	0.078	0.141
rafa/watermelon	7	0.94	0.280	0.210	0.194	0.327
reallm-labs/slime-dev	5	0.00	0.055	0.000	0.000	0.015
upup-ashton-wang-usc/Tina	1	1.31	0.293	0.196	0.191	0.414

Table 17: Pool-trained baselines under the three regimes, GRID10 MAE per cell (lower is better). The zero-shot references at the bottom never use the pool and are identical across regimes. Bold: better of the two shape regressors in a column within a regime.

Method	Regime	PT/Marin	PT/MLF	PT/AI2	FT/Marin	FT/HF
Mean training shape	sweep seen	0.182	0.231	0.195	0.009	0.113
kNN on prefix shape	sweep seen	**0.105**	**0.172**	**0.193**	**0.004**	**0.110**
Mean training shape	sweep held out	0.183	0.237	0.197	**0.214**	0.110
kNN on prefix shape	sweep held out	**0.165**	**0.205**	**0.195**	**0.214**	**0.108**
Mean training shape	source held out	**0.188**	0.247	**0.208**	**0.081**	**0.054**
kNN on prefix shape	source held out	0.354	**0.216**	0.411	0.100	0.097
NCPL (in-distribution only)	sweep seen	0.192	0.152	0.234	0.005	0.082
LC-PFN	zero-shot	0.252	0.185	0.480	0.037	0.074
Claude Opus 5 · FULL	zero-shot	0.150	0.213	0.216	0.022	0.119
Last value	–	0.533	0.495	0.575	0.097	0.108

E INFORMATION CONDITIONS AND SETTING–CURVE MATCHING

This appendix gives the analyses of the semantics referenced from §4.1: the three information conditions, what in the semantics carries the gain, and the setting–curve matching task.

E.1 INFORMATION CONDITIONS

Every API model was asked in three conditions: the curve alone (CURVE), the curve with the numeric configuration (CURVE+CONFIG), and the curve with the configuration and the semantics (FULL); per-model scores are in Table 10 and paired contrasts in Tables 7 and 8. On the GRID10 pretraining and fine-tuning runs, adding the configuration moves Claude Opus 5, GPT-5.6 Sol, GPT-5.6 Luna, Gemini 3.1 Pro, and Qwen3-Next-80B by less than 0.02 (paired intervals include zero) and raises Claude Haiku 4.5 by 0.08 $[0.03, 0.14]$ and Gemini 3.8 Flash by 0.04 $[0.00, 0.08]$; adding the semantics on top helps GPT-5.6 Sol ($+0.03$ $[0.00, 0.06]$), GPT-5.6 Luna ($+0.04$ $[0.00, 0.09]$), and Claude Haiku 4.5 ($+0.06$ $[0.02, 0.11]$), and costs Claude Opus 5 0.06 $[0.02, 0.09]$, Gemini 3.8 Flash 0.08 $[0.03, 0.13]$, Muse-Glimmer-30B 0.09 $[0.03, 0.16]$, and DeepSeek-V3.2 0.30 $[0.13, 0.47]$.

The gain concentrates where the prefix misleads. We derived schedule descriptors for every run from its configuration and source facts (an LLM check by Claude, which re-derived every descriptor of 30 random runs from their configurations, agreed on all of them) and split runs by whether warmup occupies most of the prefix. Such prefixes mislead every method: on pretraining, the curve-only error ratio is higher where warmup dominates (LC-PFN 0.443 against 0.321, GPT-5.6 Sol 0.461 against 0.333, Gemini 3.1 Pro 0.606 against 0.331). There, the paired ratio of FULL to CURVE error is 0.64 $[0.52, 0.78]$ for Claude Opus 5, 0.71 $[0.61, 0.82]$ for Gemini 3.1 Pro, and 0.72 $[0.59, 0.88]$ for GPT-5.6 Luna, against 1.32, 1.11, and 0.94 elsewhere; GPT-5.6 Sol gains in both groups (0.91 and 0.82). No pretraining or fine-tuning run reports the shape of its schedule and no RL run reports any schedule field, so the effect of a late cooldown cannot be tested, and because the groups are nearly collinear with source, these comparisons are within stage and associational.

The semantics act as a cue to the kind of run. On 100 fixed questions, replacing the semantics by those of another run from the same project (the semantics placebo) leaves the error unchanged (ratios 0.98 and 1.03), even when the learning rate, model size, or budget differs by more than a factor of two, and for GPT-5.6 Luna the task description alone recovers the whole gain.

E.2 SETTING–CURVE MATCHING

Each matching question takes two runs from the same sweep, shows both curves and both settings with run identities hidden and the reported parameter values retained, and asks which setting produced which curve. We score 340 pairs: the 150 pretraining and 70 fine-tuning main pairs of a 330-pair matching release that allows at most one reuse of any curve endpoint within a set (Appendix B), and the 120 pairs of the 96-run part of the RL set. Each pair is asked in both orders, which are averaged into one score per pair, and in two conditions (CURVES+SETTINGS and FULL), so a stage's n in Table 18 counts pair–condition rows (chance 0.5), and uncertainty is aggregated by pair, never by view. No run appears in two of the scored pairs, but pairs drawn from one sweep are not independent, and aggregating by pair does not remove that dependence: intervals resample recorded lineage components on pretraining and fine-tuning (140 components for the 150 pretraining pairs, 20 for the 70 fine-tuning pairs) and source projects on RL, so two pairs from one sweep that fall in different components are resampled as independent, and the pretraining and fine-tuning intervals may be optimistic. Table 18 gives the accuracy by model and stage; five models, among them the open-weight Kimi K3, reach 0.82–0.86 on the unweighted mean over the three stages. Three models are at chance in every stage: the open-weight DeepSeek-V3.2 and Qwen3-Next-80B, and the closed Claude Haiku 4.5. Matching by forecasting, that is, forecasting each curve under each setting and keeping the closer assignment, is at chance from a 20% prefix for the same models (GPT-5.6 Sol 0.49, Claude Opus 5 0.47): swapping the setting moves GPT-5.6 Sol's forecast by a seventh of the gap between the two curves, in the right direction half the time, whereas with no prefix the shift covers the gap and the direction is right 0.68 $[0.62, 0.74]$ of the time (266 pairs, no two sharing a run). The settings easiest to match directly (model size, initial checkpoint, RL budget: 96–100%) are the ones a forecast from a prefix does not recover, and the learning rate is hard both ways; accuracy is lowest when a single kind of setting separates the two runs and highest when four do. On speedrun pairs that

Table 18: Setting–curve matching accuracy per pair, both information conditions pooled (chance 0.5), averaged over the three stages and per stage, and the rate at which both orders of a pretraining pair are answered correctly (chance 0.25). n counts pair–condition rows: 150 pretraining, 70 fine-tuning, and 120 RL pairs, each in two conditions. The supervised row uses different protocols: its pretraining entry is trained on our pool and evaluated leave-project-out (150 pairs; 0.68 $[0.61, 0.75]$), its RL entry is trained on release pairs from other sources and evaluated leave-source-out (0.59 $[0.50, 0.66]$); no supervised variant exceeded chance on fine-tuning, where 52 of the 70 pairs differ in the initial checkpoint.

Model	Avg.	PT (n=300)	FT (n=140)	RL (n=240)	Both orders, PT
Claude Opus 5	**0.859**	0.775	**0.893**	**0.91**	0.614
Gemini 3.1 Pro	0.853	**0.838**	0.850	0.87	0.639
GPT-5.6 Sol	0.847	0.808	0.871	0.86	**0.647**
Kimi K3	0.834	0.798	0.821	0.88	0.525
Gemini 3.8 Flash	0.816	0.723	0.879	0.85	0.646
GPT-5.6 Luna	0.686	0.567	0.671	0.82	0.279
DeepSeek-V3.2	0.515	0.495	0.525	0.53	0.201
Claude Haiku 4.5	0.500	0.487	0.493	0.52	0.185
Qwen3-Next-80B	0.498	0.497	0.464	0.53	0.181
Ridge on config differences	–	0.68	–	0.59	–

Table 19: Setting–curve matching as four-way multiple choice in both directions: accuracy with a 95% interval, chance 0.25; invalid answers are scored as a uniform guess.

Model	one setting, four curves	one curve, four settings
Claude Haiku 4.5	0.252 [0.188, 0.320]	**0.339** [0.305, 0.372]
Claude Opus 5	0.319 [0.275, 0.360]	0.333 [0.303, 0.365]
Gemini 3.1 Pro	0.285 [0.250, 0.324]	0.329 [0.304, 0.354]
Gemini 3.8 Flash	0.306 [0.257, 0.355]	0.299 [0.265, 0.337]
GPT-5.6 Sol	0.262 [0.206, 0.323]	0.277 [0.244, 0.313]
GPT-5.6 Luna	0.245 [0.202, 0.292]	0.275 [0.245, 0.305]
DeepSeek-V3.2	0.275 [0.241, 0.313]	0.262 [0.236, 0.288]
Qwen3-Next-80B	0.233 [0.193, 0.276]	0.235 [0.221, 0.250]

differ in how an optimizer or schedule is implemented every model is at chance from the settings alone, and a diff of the training scripts lifts GPT-5.6 Sol to 0.88 and Gemini 3.1 Pro to 0.85; varying the diff budget over 3k, 6k, and 8k characters on the same 50 pairs has no detectable effect.

Recast as four-way multiple choice (Table 19), every model falls toward chance in both directions (maximum 0.339, six of sixteen intervals above 0.25; Claude Opus 5 passes the schema on only 285 of 434 forward questions), which suggests that the models compare two candidates rather than evaluate one.

Table 20: Observed share of the run: the same 100 questions at observed shares from 0 to 50% of the budget; MAE relative to the last value at the standard 20% (geometric mean over five cells; lower is better; Figure 3a plots $1 - R$). NCPL is the retrained checkpoint, reading ten points resampled from the observed prefix; Muse-Glimmer-30B answers 88 of the 100 questions at 0%.

Method	0	2%	5%	10%	15%	20%	50%
Last value	–	5.99	2.80	1.77	1.30	1.00	0.37
LC-PFN	–	–	2.13	0.82	0.69	0.52	0.14
GPT-5.6 Sol · FULL	1.32	1.15	0.90	0.46	0.47	0.39	0.14
Claude Opus 5 · FULL	1.57	1.10	0.68	0.48	0.41	0.38	0.15
Muse-Glimmer-30B · FULL	4.41	2.12	0.87	0.82	0.60	0.60	0.23
NCPL (retrained)	–	–	1.79	1.16	0.84	0.65	0.49

F OBSERVED SHARE, INPUT WINDOW, AND FAILURE MONITORING

This appendix gives the tables and protocols behind §4.2: the observed share of the run (Figure 3a), the granularity of input and output (Figure 3b), and failure monitoring (Figure 3c).

F.1 OBSERVED SHARE OF THE RUN

Table 20 scores the same 100 pretraining and fine-tuning questions (20 per cell) at observed shares from 0 to 50% of the budget, relative to the last value at the standard 20%. Without any curve, GPT-5.6 Sol (1.32) is better than the mean training curve of the cell (1.53), Claude Opus 5 (1.57) is not, and Muse-Glimmer-30B is far worse (4.41), whereas the released NCPL checkpoint, which uses only the configuration, reaches 0.46 on the one cell it can score: without an observation, the scale of the loss must come from similar runs. The LLMs miss the level but not the order: on a separate set of 100 questions (20 per cell) that asks for the whole curve with no prefix, the final values GPT-5.6 Sol predicts order the 20 runs of each cell with Spearman correlation 0.61–0.82 (from 0.61 $[0.22, 0.86]$ on Hugging Face fine-tuning to 0.82 $[0.52, 0.97]$ on MLFoundations pretraining). From 10% of the budget Claude Opus 5 and GPT-5.6 Sol lead every curve-only method and the retrained NCPL, which first beats the 20% last value at a 15% share; from 20% to 50% they improve by a further factor of 2.5, part of which comes from the shorter horizon that remains to be forecast.

F.2 FAILURE MONITORING

We index 358 runs whose metrics show a failure (KL blow-up, reward collapse, unrecovered loss jump, or non-finite values), and match 100 of them, each at its failure point x_c, with two healthy runs each. Two protocols use these 300 runs.

Lead time (Figure 3c, Table 21). Each run is shown a window 10% of the budget wide, holding everything logged in it, that ends a lead g of 2, 5, or 10% of the budget before x_c; a healthy run is shown the same absolute window as its failed partner, and a monitor returns the probability that the run will fail. Groups whose failure comes too early for the window drop out, leaving 300, 291, and 258 runs (100, 97, and 86 failures) at the three leads. AUROC is computed over runs. The LLMs read the window, including the logged auxiliary series, with the run's context and return a failure probability; LC-PFN and NCPL score a run by the largest adverse change their median forecast predicts over the next 10% of the budget, relative to the observed level; and the numeric detectors are trained leave-one-source-out.

Online monitoring (Table 22). Each run is visited at up to four checkpoints (the first and last admissible points of a 10, 20, …, 90% grid and two equally spaced between them; 1,101 checkpoints over the 300 runs, a failed run only before its failure point), and each visit discloses everything logged up to the checkpoint and nothing after it. Every monitor is scored on these checkpoints. A monitor may raise one alarm per run, and its threshold is fitted on held-out sources to allow one false alarm per five healthy runs. In this online protocol, at that false-alarm rate and excluding a KL-divergence threshold, which reads the labeling criterion of 38 of the 100 failures, no monitor catches half of the failures: the best is boosted trees on curve statistics, which catch 0.41 $[0.32, 0.51]$ of them with a median lead time of 0.60 of the budget, save 0.20 $[0.15, 0.26]$ of the budget per failed run (the span between the alarm and the failure point, zero for a missed failure), raise a false alarm on 0.205

Table 21: Failure detection by lead time (Figure 3c): AUROC over runs of a monitor shown a 10%-wide window ending a lead g before the failure point (chance 0.5; brackets are 95% intervals). Numeric detectors are trained leave-one-source-out; the trees are the $g \leq 0.10$ part of Figure 8.

Monitor	$g = 2\%$	$g = 5\%$	$g = 10\%$
Runs (failures)	300 (100)	291 (97)	258 (86)
Logistic, primary metric	0.65	0.62	0.58
Boosted trees on curve statistics	0.72 [0.66, 0.78]	0.66 [0.60, 0.73]	0.65 [0.57, 0.72]
LC-PFN	0.45 [0.37, 0.53]	0.46 [0.39, 0.54]	0.48 [0.41, 0.56]
NCPL (retrained)	0.43 [0.36, 0.50]	0.45 [0.37, 0.53]	0.47 [0.39, 0.55]
Claude Opus 5	0.73 [0.68, 0.78]	0.64 [0.58, 0.69]	0.61 [0.55, 0.68]
GPT-5.6 Sol	0.73 [0.68, 0.78]	0.67 [0.62, 0.73]	0.67 [0.62, 0.74]
GPT-5.6 Luna	0.74 [0.69, 0.79]	0.68 [0.63, 0.73]	0.65 [0.59, 0.71]
Muse-Glimmer-30B	0.68 [0.62, 0.75]	0.61 [0.55, 0.66]	0.57 [0.50, 0.64]

Table 22: Online failure monitoring on 100 failed and 200 healthy runs, each visited at up to four checkpoints between 10% and 90% of the budget (1,101 checkpoints), at one false alarm per five healthy runs. Lead time, compute saved (mean over failed runs of the budget between alarm and failure point), and healthy stopped (mean over healthy runs of the budget after a false alarm) are fractions of a run's budget; brackets are 95% cluster-bootstrap intervals. A KL-divergence threshold (0.51 $[0.42, 0.61]$) is omitted because 38 of the failures are labeled by a KL criterion.

Monitor	Failures caught	Median lead time	Compute saved	Healthy stopped
Loss spike $> k\sigma$	0.17 [0.10, 0.25]	0.08	0.04	0.12
Gradient-norm rule	0.30 [0.21, 0.39]	0.17	0.11	0.16
Logistic, next 10%	0.35 [0.26, 0.44]	0.10	0.09	0.08
Boosted trees on curve statistics, rest of run	**0.41** [0.32, 0.51]	0.60	0.20	0.17
GPT-5.6 Luna · FULL	0.22 [0.14, 0.30]	0.06	0.04	0.16
Claude Opus 5 · FULL	0.29 [0.21, 0.38]	0.08	0.07	0.16
GPT-5.6 Sol · FULL	0.36 [0.27, 0.46]	0.09	0.07	0.15

$[0.15, 0.26]$ of the healthy runs, and forgo 0.17 $[0.13, 0.22]$ of the budget per healthy run; the LLMs with the FULL context catch 0.22–0.36. These recall figures belong to the online protocol only; the lead-time AUROCs of Table 21 rank runs at a fixed lead without a threshold, and there the best LLM and the boosted trees are above chance at every lead (at $g = 10\%$, GPT-5.6 Sol 0.67 $[0.62, 0.74]$ and the trees 0.65 $[0.57, 0.72]$). Stacking an LLM's failure probability onto the numeric features changes AUROC by 0.013 $[-0.011, 0.037]$ for GPT-5.6 Sol and 0.018 $[-0.008, 0.047]$ for Claude Opus 5.

Figure 8 extends the lead-time protocol for the best numeric detector from $g = 0$ to $g = 0.30$: the pooled AUROC stays flat beyond $g = 0.15$, KL blow-ups fall to chance beyond a 10% lead (0.58, 0.43, 0.37 at $g = 0.10$, 0.15, 0.20), and reward collapses decay from 0.81 to 0.53. For an LLM monitor, recency matters more than coverage: for GPT-5.6 Sol and GPT-5.6 Luna, the last 10% before a 2% lead gives 0.73–0.74, the first and last 5% of the run 0.69, and the first 10% or the configuration alone 0.53–0.56. Across 90 coded alarms, false alarms cite the configuration in 77% of cases against 50% for correct early alarms, which instead cite the shape of the curve and the gradient norm.

F.3 GRANULARITY OF INPUT AND OUTPUT

We re-issued 95 questions of the frozen panel at input windows of 2, 1, 0.5, and 0.2% of the budget and output grids of 10, 5, and 2%, always scoring the common points $x \in \{0.3, \ldots, 1.0\}$ against unchanged targets; the 86 questions whose raw logs are dense enough for a 0.2% window form the fixed set of Table 23 and Figure 3b. A finer input window leaves GPT-5.6 Sol's per-question error unchanged (paired ratio 1.00 $[0.84, 1.20]$ between the 2% and 0.2% windows); the table's R moves from 0.335 to 0.300 because it averages questions within each cell and then weights the five cells equally, whereas the paired ratio weights questions equally. Claude Opus 5 stays between 0.29 and 0.35 at every window, whereas Muse-Glimmer-30B improves from 0.54 at the 2% window to 0.34 at 0.2%. Among the curve methods only Chronos-2 gains (paired 0.72 $[0.65, 0.81]$), while

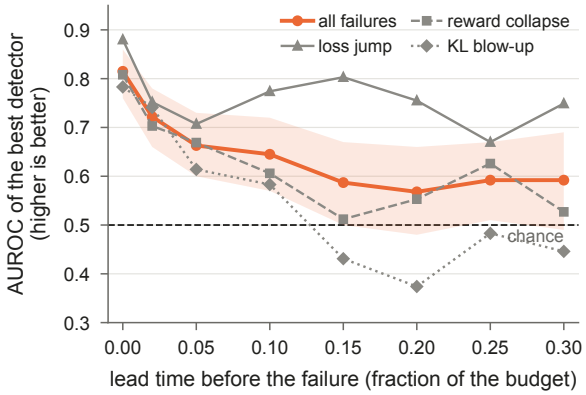

Figure 8: Beyond a lead of about a tenth of the budget, only unrecovered loss jumps remain detectable. AUROC of the best numeric detector (boosted trees on curve statistics of every logged series, trained leave-one-source-out) in a 10%-wide window ending a lead time g before the failure, pooled and by failure type (chance 0.5; band: 95% interval of the pooled line). The pooled AUROC falls from 0.82 at $g = 0$ to 0.59 at $g = 0.15$; loss jumps stay at 0.75 at $g = 0.30$.

Table 23: Error ratio R (lower is better) on the common prediction points as the input window shrinks (86 questions, the same in every column; Figure 3b plots $1 - R$). Each method is scored on its valid answers (Claude Opus 5 84–86, others 85–86). NCPL reads ten window means by construction and has one entry.

	input window			
method	2%	1%	0.5%	0.2%
last value	1.000	1.000	1.000	1.000
AIC ensemble	0.595	0.852	0.951	0.892
LC-PFN	**0.352**	0.411	0.436	0.409
Chronos-2	1.223	1.020	1.077	0.900
TimesFM-2.5	1.709	1.993	2.463	2.430
GPT-5.6 Sol (full)	0.335	**0.249**	0.296	0.300
GPT-5.6 Sol (curve only)	0.395	0.418	0.306	0.368
GPT-5.6 Luna (full)	0.360	0.396	0.335	0.378
Claude Opus 5 (full)	0.346	0.330	0.290	0.304
Muse-Glimmer-30B (full)	0.538	0.516	0.585	0.343
NCPL (retrained)	0.604	–	–	–

TimesFM-2.5 (1.73 [1.43, 2.15]) and the AIC ensemble (1.45 [1.24, 1.71]) get worse; NCPL reads ten window means by construction, so its single entry is repeated across the windows of Figure 3b. Question sets and averaging differ from Table 20 (86 against 100 questions; per-question against per-cell ratios), so its entries at the 2% window differ from the 20% column there. A finer output grid leaves LLM error at the common points unchanged (six paired contrasts between 0.88 and 1.02, every interval covering 1), but on a 40-point grid the schema-valid rate falls to 0.77 for GPT-5.6 Sol and 0.47 for Claude Opus 5, every failure a truncation at the 4,096-token answer bound, while GPT-5.6 Luna stays at 0.99.

G TEAREADER: DETAILS

This appendix gives the recipe, training data, training pools, untrained baselines, three-stage averages, and ablations of TEAREADER (§4.3).

G.1 RECIPE AND TRAINING DATA

- **Backbone & adapter.** Qwen3.5-2B; LoRA r=64 on the attention and MLP projections (q, k, v, o, gate, up, down), lr 2×10^{-4}.
- **Target.** Each query point as a *ratio* to the last observed value, 4 decimals, as text.
- **Prompt.** The re-gridded curve prefix plus the numeric reported configuration; no description, no code, no retrieved runs.
- **Data.** 7,724 examples from 918 runs, with every test lineage and test sweep removed; add three truncated views per example only if short observations are part of the target use.
- **Schedule.** 1,000 optimizer steps, effective batch 8, cosine, 3% warmup, max length 8,192, loss on answer tokens only (≈1.2 GPU-hours of training).
- **Decoding.** Greedy; three training seeds of the reference differ by up to 0.06 on pretraining and 0.08 on fine-tuning runs of the frozen test.

The reference pool has 918 runs and 7,724 training examples: 578 runs from the strict pool of Appendix B, from which every test lineage component and test sweep family is removed (1,463 runs), and 340 public Hugging Face fine-tuning and RL runs admitted under the same rules; by stage it holds 218 pretraining, 52 synthetic-task, 280 fine-tuning or preference-tuning, and 368 RL runs. An audit of the 918 training runs finds no test run, no run from a non-Marin test project, no test sweep family, and no test lineage component; against the community set they share no repository and no lineage, but 107 of the 232 community runs share an author and 61 a model family with a training run. The target representation decides whether training works: trained to emit raw metric values the reader is worse than LC-PFN, and writing each target as a ratio $y(x)/\bar{y}_{10}$ to the last observed value lowers the error on the Marin fine-tuning cell from 0.087 (raw) to 0.054 (difference) and 0.035 (ratio). Quantiles written as text are too wide (coverage 0.89–0.93 at a nominal 0.80); a separate quantile head trained with pinball loss restores coverage to 0.79 at a 4–17% cost in point error.

G.2 TRAINING POOLS AND BACKBONES

Table 24 gives the reader under the recipe on four training pools and two backbones, reading the configuration alone (config) or the configuration and the semantics condensed into one paragraph (config + semantics), with three paired training seeds each. The 2,641- and 3,002-run pools each add runs of one kind to the previous pool; the 8,727-run pool removes 22 runs of the 3,002-run pool before adding its own (below), so the pools are not nested, and the 9,112-run pool adds 385 runs to the 8,727-run pool. The 2,641-run pool adds 1,723 public experiment-tracker runs, mostly pretraining (Marin ablations, optimizer benchmarks, and scaling sweeps), which are not isolated by project (53 of them, and 546 more in the 8,727-run pool, come from a Marin project that also supplies test runs of other sweeps). The 3,002-run pool adds 361 Hugging Face Hub runs (356 RL, 5 fine-tuning), admitted under a family rule that treats a configuration default shared by unrelated owners as no relation; these are the only RL runs added to any pool. The 8,727-run pool adds 5,747 public runs, almost all pretraining, and removes 22 Marin pretraining runs (180 training examples) that had entered with the 2,641-run pool and were carried into the 3,002-run pool. None of the 22 is a test run, but once the relation records collected for the 8,727-run pool were added, each was linked to a test run through a chain of sweep-family records, so the sweep-family isolation of §3.2 would have excluded it; the 918-, 8,727-, and 9,112-run pools contain none of them. The readers trained on the 2,641- and 3,002-run pools, that is, every point of Figure 4 at those two pool sizes (both backbones, both inputs, all three stages), are therefore not strictly isolated from the test runs, whereas the points at 918 and 8,727 runs satisfy that rule. No added run has an audited description, so its semantics paragraph is rendered from its numeric configuration alone; the semantics variant therefore carries new textual evidence only for the 918 reference runs. Three caveats qualify the table: the 9B and 2B readers were trained on different clusters, so size is confounded with hardware and software; the rows of the 2,641- and 3,002-run pools include the 22 runs above; and community-run scores here

Table 24: Reader by training pool, backbone, and input (skill score S, higher is better). PT and FT: the 150-run GRID10 test sets; RL: the 96-run part of the RL set (both grids); Avg.: unweighted mean of PT, FT, and RL; PT+FT: the five pretraining and fine-tuning cells pooled; held-out sources and held-out families: validation splits; wild: the community set with FULL prompts. Trained readers: mean of three training seeds. Untrained, benchmark: the benchmark prompt, CURVE+CONFIG for 2B (no RL answers) and FULL for 9B. Untrained, reader: the reader's own prompt and ratio target without training, scored on valid answers (2B config / config + semantics: 131 / 44 pretraining and 149 / 111 fine-tuning answers of 150; 9B: 122 / 110 and 150 / 131; RL: 51–82 of 82 scoreable runs). LLM rows use the FULL prompt. †This pool adds 385 Hugging Face runs of the same kind as the community set (no shared repository; 43 share an author, 19 a model family), so its community-run score is not a transfer test.

Pool	Model, input	PT	FT	RL	Avg.	PT+FT	Held-out sources	Held-out families	Wild
918 (reference)	2B, config	0.688	0.607	−0.332	0.321	0.659	0.680	0.237	0.267
918 (reference)	9B, config	0.699	0.604	−0.085	0.406	0.664	0.654	0.245	0.327
918 (reference)	9B, config + semantics	0.701	0.603	−0.031	**0.424**	0.665	0.622	0.244	0.289
2,641	2B, config	0.678	0.495	−0.489	0.228	0.615	0.757	0.223	0.292
2,641	2B, config + semantics	0.700	0.569	−0.770	0.166	0.653	0.796	0.249	0.255
2,641	9B, config	0.708	0.527	−0.291	0.315	0.648	0.722	0.233	0.266
2,641	9B, config + semantics	0.691	0.595	−0.186	0.367	0.659	0.726	0.239	0.307
3,002 (+356 RL)	2B, config	0.686	0.592	−0.254	0.341	0.652	0.778	0.222	0.298
3,002 (+356 RL)	2B, config + semantics	0.699	0.616	−0.538	0.259	0.668	0.784	0.219	0.310
3,002 (+356 RL)	9B, config	0.679	0.590	−0.320	0.316	0.646	0.713	0.223	0.282
3,002 (+356 RL)	9B, config + semantics	0.711	0.639	−0.306	0.348	0.685	0.729	0.222	0.286
8,727	2B, config	0.689	0.602	−0.269	0.341	0.659	0.810	0.271	0.292
8,727	2B, config + semantics	0.711	0.556	−0.413	0.285	0.660	0.801	0.229	0.266
8,727	9B, config	0.711	0.675	−0.645	0.247	0.697	0.790	0.217	0.229
8,727	9B, config + semantics	0.725	0.705	−0.322	0.369	0.717	0.779	0.228	0.297
9,112 (+385 wild-class)	2B, config	0.714	0.599	−0.272	0.347	0.673	0.809	0.290	0.288†
Untrained, benchmark	2B, CURVE+CONFIG	0.584	0.181	–	–	–	–	–	–
Untrained, benchmark	9B, FULL	0.375	0.470	0.032	0.293	–	–	–	–
Untrained, reader	2B, config	−12.1	−1.18	−3.40	–	–	–	–	–
Untrained, reader	2B, config + semantics	−5.74	0.03	−3.74	–	–	–	–	–
Untrained, reader	9B, config	−19.5	−1.15	−3.11	–	–	–	–	–
Untrained, reader	9B, config + semantics	−15.4	−0.52	−3.06	–	–	–	–	–
LLM	Claude Opus 5	0.697	0.559	0.385	0.547	–	–	–	–
LLM	GPT-5.6 Sol	0.659	0.653	0.312	0.542	–	–	–	–
LLM	Muse-Glimmer-30B	0.350	0.564	0.108	0.341	–	–	–	–

are over all 232 runs with the FULL prompts on both grids (464 views), not over the core-211 cohort of Figure 2.

The recipe was chosen on the reference pool after its frozen, RL, and community-run scores had been seen (Table 26), so the reference-pool results and the test columns of Tables 26 and 28 are exploratory. The three-seed runs of Table 24 on the larger pools and with the 9B backbone were trained afterwards with the recipe unchanged, and every configuration of that matrix is reported; they involve no further choice made on test scores, except that the 9,112-run arm was added after the community-run scores of the other pools had been seen, but they inherit the recipe and are not an independent confirmation of it.

G.3 UNTRAINED BACKBONES AND THREE-STAGE AVERAGES

The untrained backbones are scored in two ways (Table 24). Under the benchmark prompts of §3.1, the untrained Qwen3.5-2B is scored with the CURVE+CONFIG prompt, the input closest to the reader's, and reaches $S = 0.58$ on pretraining and 0.18 on fine-tuning (143 and 146 of 150 answers valid; it was not run on RL under that prompt); the untrained Qwen3.5-9B has no CURVE+CONFIG answers and is scored with the FULL prompt, reaching 0.38, 0.47, and 0.03 on pretraining, fine-tuning, and RL. Under the reader's own prompt, neither backbone produces usable forecasts before training, and the trained-minus-untrained contrasts of Table 8, which use this matched prompt, are positive for every backbone, input, seed, and stage. The semantics raise the 9B three-stage average on every pool and lower the 2B average on the three larger pools.

Table 25: Reader input and scale on validation (error ratios against the last value; lower is better). *Combined* is the geometric mean of the point and pinball-loss ratios. Left: input at 2B. Right: paired ratio of errors when the semantics paragraph is added to the configuration, per example with 95% interval, on pretraining and fine-tuning examples.

Input (2B)	Point	Pinball	Combined	Scale	+semantics / config
Curve only	0.524	1.686	0.940	0.8B	1.03 [0.95, 1.12]
+ numeric configuration	0.523	0.635	0.576	2B	1.03 [0.95, 1.10]
+ schedule facts	0.502	0.647	0.570	9B	1.02 [0.98, 1.07]
+ semantics paragraph	0.491	0.636	0.559	Qwen3 1.7B / 8B	1.06 [1.01, 1.13] / 1.07 [1.02, 1.13]
+ one related run	0.486	0.933	0.673	Qwen3 14B	0.89 [0.76, 1.00] (readings disagree)

G.4 Ablations

Tables 25, 26, 27, and 28 give the ablations on validation runs, scored by the geometric mean of the point and pinball-loss error ratios against the last value. Because cells differ greatly in size, every contrast is read three ways (macro average over all cells, macro average over cells with at least 20 examples, and the per-example paired ratio), and a conclusion is reported only when two readings agree. The numeric configuration is the only large step on the input axis, and it improves the intervals rather than the median (point ratio $0.524 \to 0.523$, pinball ratio $1.69 \to 0.64$). On validation at 2B, schedule facts have no effect, and neither has the semantics paragraph added to the configuration in Table 25 (paired ratio $1.03\ [0.95, 1.10]$); the run-level semantics variant of Table 27, trained on the relaxed pool (Table 5), is worse ($1.41\ [1.19, 1.71]$), and the RL cost of the semantics at 2B appears on the test sets (Table 24). Replacing each example's schedule quantities with those of another run changes nothing ($1.00\ [0.91, 1.12]$). The pooled validation gain from the semantics at 9B (0.91 $[0.77, 1.03]$ against 1.12 $[1.02, 1.22]$ at 0.8B) comes from RL examples, whose semantics paragraph adds only numeric fields: without them the ratio is 1.03, 1.03, and 1.02 at 0.8B, 2B, and 9B. A Qwen3 ladder (1.7B, 8B, 14B; one seed per rung) shows no gain from scale, so the 14B threshold reported for context-aided forecasting (Ashok et al., 2026) is not reproduced. Frontier-model forecasts are not used as a reader input: in a predeclared validation check, the mean of three such forecasts has 0.48 $[0.36, 0.64]$ of the reader's error on 41 held-out-source runs but 1.49–1.69 times its error on 74 held-out-family runs, so no fixed combination beats both.

Table 26: Ablation of the TEAREADER recipe (top row): Qwen3.5-2B, LoRA r=64, 1,000 steps on the *strict* pool (test lineage components and sweep families removed), ratio target, curve + numeric configuration, greedy decoding. Most rows change one component; the three-thousand-step and full-fine-tuning rows change the number of steps, the data mix, or the optimizer together. The recipe was chosen after these frozen, RL, and wild scores were seen (Appendix G); the test columns of this table are therefore exploratory, whereas the three-seed runs on the larger pools and with the 9B backbone (Table 24) were trained after the recipe was fixed. Lower is better; 1.0 = no skill over repeating the last observed value. Validation (left block) is read three ways—macro over all cells, macro over cells with $n \geq 20$ (post hoc), and the paired per-example ratio—and a verdict is given only when two readings agree. GRID10, GRID20, RL, and wild (right block) give the error ratio R on the test sets. The RL column uses one scoring pipeline for every variant (the 96-run part of the RL set, 82 scoreable runs) and is called *worse* or *better* only when a paired run-cluster bootstrap interval excludes zero; empty RL cells were never generated. Frozen-test and wild columns are called only outside ± 0.02, the difference between two training seeds of one variant; this is a heuristic, not a confidence interval. Confidence intervals are over validation *families* (over run clusters for RL and wild), *not* over training seeds, which move single contrasts by about $\pm 20\%$. Variants trained on the relaxed pool are in Table 27.

	Validation				Test sets				
Ablation	all	$n \geq 20$	paired vs. ref. [CI95]	verdict	G10	G20	RL	wild	test verdicts
Recipe									
reference recipe: 1k steps, LoRA r=64, curve + numeric config, strict pool	0.591	0.476	– (ref.)	–	0.325	0.312	1.31	0.717	–
Training length									
3k steps (same data)	0.692	0.400	0.852 [0.671, 1.022]	better (ns)	0.337	0.364	–	–	frozen =
Observation mix									
short-observation mix (3 truncated views), 1k steps	0.640	0.516	1.079 [1.033, 1.130]	worse	–	–	–	–	–
short mix + 3k steps	0.555	0.443	–	better	0.325	0.367	1.33	0.797	frozen =, RL ns, wild worse
short mix + 3k steps, seed 2	0.576	0.409	–	better	0.344	0.316	1.68	0.806	frozen =, RL worse, wild worse
Optimization									
full fine-tune (fp32 master weights), short mix, 3k steps	0.554	0.398	0.857 [0.766, 0.951]	better	0.321	0.356	1.42	0.901	frozen =, RL ns, wild worse
Data distribution									
Marin down-sampled to 1/3 (cell balance)	–	–	–	n/a	0.346	0.309	1.30	0.707	frozen worse, RL ns, wild =
no Marin in training	0.739	0.666	–	worse	0.334	0.404	1.09	0.696	frozen =, RL better, wild better
Marin only (1,296 examples)	0.552	0.523	–	mixed	0.425	0.397	1.88	1.127	frozen worse, RL worse, wild worse
Supervision									
inline rationale before the numbers (9% of examples)	0.756	0.544	–	worse	–	–	–	–	–

Table 27: Recipe variants trained on the relaxed pool (Table 5) and scored on validation only, against the recipe trained on the same relaxed pool (not paired with the strict-pool reference of Table 26). Columns and verdict rule as in Table 26; lower is better.

Variant	all cells	$n \geq 20$	paired vs. ref. [CI95]	verdict
Recipe variants				
+ run-level semantics	0.901	0.740	1.411 [1.186, 1.713]	worse
semantics placebo (same cell) ÷ true semantics	1.197	0.747	1.041 [0.986, 1.099]	worse (ns)
full fine-tune (fp32 master weights, lr 10^{-5})	0.560	0.434	0.990 [0.844, 1.159]	better (ns)
LoRA r=256	0.590	0.472	0.954 [0.860, 1.044]	better (ns)
25% of the training runs	0.746	0.527	1.241 [1.076, 1.453]	worse
50% of the training runs	0.697	0.480	1.085 [0.994, 1.203]	worse (ns)
Input and data variants				
+ schedule facts as numbers	0.570	0.502	0.951 [0.781, 1.134]	worse (ns)
schedule facts shuffled across runs (placebo)	0.652	0.499	1.001 [0.913, 1.119]	null
+ 1 retrieved related run (train and inference)	0.673	0.498	1.238 [1.113, 1.383]	worse
1,058 supplementary same-source examples removed	0.576	0.492	1.061 [1.001, 1.130]	worse

Table 28: Further reader variants against the recipe, paired on common valid examples (error ratio of point forecasts, variant over recipe; below 1 is better; 95% intervals over validation families). *rag*: two or three BM25-retrieved training runs from other sweep families in the prompt at training and inference; *telemetry*: learning rate, gradient norm, KL, entropy, and response length before the cutoff; *damping*: the predicted change scaled per stage after training. *Clean*: sweep-held-out runs without a near-copy neighbor. Frozen: error ratio R on the frozen test release, GRID10; the recipe scores 0.325.

Variant	held-out source	held-out sweep	held-out sweep, clean	frozen R
rag ($k = 3$)	1.29 [1.15, 1.47]	0.80 [0.68, 0.90]	0.98 [0.94, 1.01]	0.449
rag, same model without neighbors	1.11 [1.03, 1.21]	1.02 [0.96, 1.08]	1.00 [0.96, 1.03]	–
rag, irrelevant neighbors in training	1.16 [1.07, 1.27]	0.91 [0.84, 0.97]	0.97 [0.93, 1.02]	–
telemetry	1.15 [1.08, 1.24]	1.00 [0.95, 1.05]	0.99 [0.95, 1.03]	0.354
damping (post hoc)	–	–	–	0.327

Table 29: A database of 1,726 past runs, queried on the same 125 questions with GPT-5.6 Sol. Top left: all valid questions pooled (Figure 5 separates the two question groups); paired change in MAE against the same model without retrieval (negative is better) and the geometric mean of the ratio of the two errors, by database size, by k, and, for the full database, by quartile of the nearest neighbor's similarity. Top right: top-1 accuracy of the sweep recommender; the best label-free prior picks the value most common in the database. Bottom: MAE per cell (in each source's own units, 25 questions per cell) against the better of two regressors trained on the same runs (kNN on prefix shape and mean training shape) under the regimes of Appendix D.8; absolute levels are not comparable with Table 17. *One question: the other 24 MLFoundations questions have no run of their own sweep family in the database.

GPT-5.6 Sol arm	ΔMAE	ratio
Library 17 runs, k=3	−0.039	0.90
86 runs	−0.027	0.91
431 runs	−0.040	0.71
1,726 runs	−0.063	**0.61**
k=1	−0.063	0.75
/ k=5	/ −0.052	/ 0.63
Nearest neighbor, Q_1	+0.003	0.85
Q_2	−0.098	0.60
Q_3	−0.056	0.47
Q_4	−0.102	0.57

Recommender: 70 questions, 7 settings	
GPT-5.6 Sol + retrieved sweeps	**0.61**
GPT-5.6 Sol, configurations only	0.46
Best label-free prior	0.37
Random	0.39

Method (isolation)	FT/HF	FT/Marin	PT/AI2	PT/Marin	PT/MLF
Best regressor, leave-family-out	0.117	0.154	**0.210**	0.270	0.231
GPT-5.6 Sol, no retrieval	0.057	0.031	0.410	0.262	0.237
GPT-5.6 Sol + retrieval, leave-family-out	**0.049**	**0.027**	0.240	**0.186**	**0.182**
Best regressor, in-distribution	0.120	0.0037	0.209	0.209	0.196
GPT-5.6 Sol + retrieval, siblings allowed	**0.040**	**0.0021**	**0.150**	**0.108**	0.039*

H DATABASE OF RUNS: DETAILS

This appendix gives the protocol and the full results behind §4.4.

The database holds 1,726 training runs with their configuration, code, and lineage. For each of 125 pretraining and fine-tuning questions we embed the configuration text, retrieve the k nearest runs outside the question's sweep family, and add their complete curves to the FULL prompt of GPT-5.6 Sol; the score is the paired change in error against the same model's answer without retrieval. Smaller databases are subsamples of 1, 5, and 25% of the runs. Table 29 (top left) gives the gain by database size, by k, and by quartile of the nearest neighbor's cosine similarity; ratios for further retrievers and for GPT-5.6 Luna are in Table 9.

Questions whose own series is in the database. Leave-family-out retrieval removes the question's sweep family, but a project can group several sweep families into one coarser experiment series (for example, one study launched as several sweeps). For 24 of the 123 valid questions, all Marin pretraining, at least one of the three runs retrieved from the full database belongs to the question's series though not to its sweep family. Figure 5 draws these 24 questions hollow at every database size (the group is fixed by the full-database retrieval; in the 1, 5, and 25% subsamples, 6, 14, and 21 of them retrieve such a run) and the other 99 filled. With the full database their error ratio is 0.58 $[0.34, 0.96]$ (a 42% reduction) against 0.61 $[0.47, 0.79]$ for the other 99; with the smallest database it is 1.15 $[0.84, 1.61]$ against 0.85 $[0.72, 0.99]$, so the gain does not come from runs of the question's own series.

The gain grows with the database because the nearest neighbor becomes closer: from the smallest to the full database the mean cosine similarity of the best neighbor rises from 0.870 to 0.944, and the share of neighbors from the question's own project from 0.20 to 0.93. Retrieval leaves a heavy tail: 11–16% of questions become more than twice as bad, and neither retrieving only above a similarity threshold (chosen on half of the questions, no effect on the other half) nor listing each neighbor's

configuration differences from the question (-0.023 $[-0.067, +0.005]$ over the plain neighbors) removes it.

In the comparison with trained regressors (Table 29, bottom), every method is scored on the same questions, and "siblings allowed" means that the question's own sweep family may be retrieved. The recommender questions are built only from training-pool sweeps and ask which of the values actually tried for one of seven settings ends lowest; retrieved sweeps lower the normalized regret from 0.31 to 0.19. The recommender is perfect on the 15 depth and model-scale questions, at 51% on the 45 learning-rate questions, and at chance on the 4 that differ only in the random seed; the remaining six questions cover batch size (one), budget (two), expert count (one), and warmup (two). The pool holds no controlled sweep of schedule shape, weight decay, data mixture, or any RL setting.

I CODING AGENT: DETAILS

This appendix gives the protocol of the agent study of §4.5, which compares three arms (*single call*, *agent*, and *agent + search*), and a pilot that compared agents with single calls given the curves of related runs.

I.1 PROTOCOL OF THE AGENT STUDY

Each question becomes an offline folder with the prefix, the configuration, the description, and the code excerpts of the FULL prompt. In the *agent* arm the model can list, read, and write files in that folder and run Python in a network-isolated sandbox that mounts only the folder, for up to twenty turns; the *agent + search* arm adds literature search and public-code search, and the tool layer, not the prompt, removes every result from the run's own project (no leaked run identifier or target value was found among the returned items). The *single call* is the frozen FULL answer of §4.1. The question set is the first 100 pretraining and first 100 fine-tuning questions in a fixed task-identifier order, plus 50 RL questions, each run by Claude Opus 5 and GPT-5.6 Sol in both agent arms (1,000 attempts); the set was chosen by task identifier before any answer was seen. Six attempts were never run after a provider circuit break, and missing outcomes are not imputed. Provider errors, not the models, account for most missing answers: GPT-5.6 Sol returned a valid forecast on 81–87 of the 100 pretraining or fine-tuning questions per stage in the two agent arms (48–50 of the 50 RL questions) and Claude Opus 5 on at least 96 of 100. On average an agent attempt took 5.2–5.3 turns without search and 7.2–8.4 with search, and cost \$0.17 (Claude Opus 5) and \$0.08 (GPT-5.6 Sol) per question without search and \$0.58 and \$0.13 with it. Paired contrasts of each arm against the single call, per stage, are in Table 8.

I.2 PILOT: RELATED RUNS AGAINST AGENT BEHAVIOR

Before the agent study, a pilot on 50 pretraining and fine-tuning questions (ten per source) gave every arm the same extra information, the complete curves of three to five runs from the same project but a different sweep, so that the interface could be separated from the information (Table 30). The related curves account for the whole gain: they reduce the error of Claude Opus 5 by 24% (paired difference -0.037 $[-0.075, -0.005]$) and of GPT-5.6 Sol by 44% (-0.086 $[-0.166, -0.031]$), whereas the agent is level with the single call that holds the same curves ($+0.002$, interval containing zero), and search changes the error by -0.005 $[-0.018, +0.009]$ (Claude Opus 5) and $+0.001$ $[-0.006, +0.013]$ (GPT-5.6 Sol). Without the related curves (arm $4'$) the agent with search is no better than a plain single call. The lower half of Table 30 summarizes the logs: agents fitted a parametric law in most attempts, no agent revised a written forecast, and with the related curves the per-question errors of the agents correlate with those of the single call that holds the same curves (0.62–0.97).

Table 30: The agent pilot on 50 pretraining and fine-tuning questions. Top: MAE (46–50 answered per condition); arms 2a–4 see the same related runs; arm 3 is an agent in an external coding-agent harness, and arms 4 and 4′ are agents + search in our harness. Bottom: what the agents did, from the harness logs of the 250 pilot attempts (50 questions × 5 arm–model pairs). "Read" counts attempts that opened a related curve or a source file beyond the manifest; "fitted" counts attempts whose executed code called an optimizer or regression on a power-law, exponential, or logarithmic form, or ran the provided fitting script. [a]Arm 3 ran in a different harness whose shell calls often read several files at once, so its call counts are not comparable. [b]Not metered; estimated from token counts at list price. The single call with related curves (arm 2a) costs $0.05 per question.

Condition	Claude Opus 5	GPT-5.6 Sol
1 Single call, FULL prompt	0.152	0.192
2a + complete curves of 3–5 related runs	0.116	**0.107**
2b + related runs + training code	**0.101**	0.108
3 Agent, same files (external harness)	0.118	–
4 Agent + search (literature and code)	0.111	0.108
4′ Agent + search, no related runs	0.139	0.216

	Arm 3	Arm 4		Arm 4′	
	Claude Opus 5	Claude Opus 5	GPT-5.6 Sol	Claude Opus 5	GPT-5.6 Sol
Model turns, mean [range]	9.0 [7–12]	8.9 [6–14]	8.6 [6–13]	11.4 [6–16]	10.1 [7–15]
Tool calls per question[a]	10.3	15.7	26.1	17.0	20.9
Python executions per question	3.4	3.1	2.6	3.2	3.4
Read the related curves	50/50	48/50	50/50	–	–
Read the training code	16/50	34/50	50/50	40/50	50/50
Fitted a parametric law	47/50	33/50	46/50	47/50	50/50
Searches per question (paper / code)	–	1.2 / 1.7	1.3 / 2.4	1.4 / 4.6	1.7 / 3.5
Code searches answered by the index	–	52/87	50/86	13/231	35/171
Retrieved items shown to the model	–	686	1,115	398	399
Items dropped by the project filter	–	68	98	0	0
Changed a written forecast	0/50	0/50	0/50	0/50	0/50
Output tokens per question	6.1k	6.5k	3.8k	8.2k	3.7k
Cost per question ($)	≈0.55[b]	0.70	0.23	0.98	0.24
Median wall clock (s)	80	121	95	135	60
Correlation of per-question error with arm 2a	0.62	0.96	0.97	0.69	0.51

J PROMPTS AND A FULL EXAMPLE

This appendix gives worked examples for the tasks of §3.1 (one complete question with its FULL input and the answer, three forecasts per cell, matching rationales, and one agent transcript) and the prompts of the prediction, matching, and agent tasks.

J.1 FULL EXAMPLES

One complete question and answer. The pretraining run of Figure 1 (case `92793c43c27fab0a87980503`, Marin, GRID10) as Claude Opus 5 saw it in the FULL condition, and the answer it returned. The request is the prediction prompt of Appendix J, a line `INPUT:`, and the input below as one line of compact JSON with sorted keys (3,530 bytes); only the layout is changed for display (the code excerpt, one JSON string in the request, is printed after the object), and every value is byte-exact.

```
<PREDICTION_PROMPT>
INPUT:
{"axis": "reported log index divided by configured update budget; not completed-update count",
 "case": "92793c43c27fab0a87980503",
 "context": {
   "code_context": [{"binding": "project_level_example_not_run_revision",
      "code": "<CODE, below>", "language": "python", "line_end": 337, "line_start": 322,
      "source_sha256": "ae88a8d1e95b7bcd75710bb2913fbcadf429e17257a8ef3fec23f815c49a0024",
      "source_url": "https://github.com/marin-community/marin"}],
   "interpretation_rule": "Reported per-run configuration takes precedence over source example
      defaults. Project-level code is explanatory context, not proof of execution. Missing
      information is unknown, not zero.",
   "known_unknowns": ["Readable training data mixture and tokenizer identity are not linked
      for this run.", "This source example is not bound to a recorded run revision."],
   "reported_configuration": {"model.hidden_dim": 512, "model.max_seq_len": 8192,
      "model.num_heads": 8, "model.num_layers": 1,
      "optimizer.learning_rate": 0.0028575255750067892, "optimizer.warmup": 0.01,
      "optimizer.weight_decay": 0.1, "planned_step_budget": 1058, "trainer.log_every": 1,
      "trainer.trainer.num_train_steps": 1058, "trainer.trainer.seed": 0,
      "trainer.trainer.train_batch_size": 64, "trainer.z_loss_weight": 0.0001},
   "source_facts": ["Do not assume that train/loss is pure cross-entropy: project source can
      include log-sum-exp and router auxiliary penalties."],
   "task_description": "Language-model pretraining candidate from the Marin MoE experiment
      collection. Forecast the reported training objective loss using the early trajectory and
      the reported architecture, optimizer and training-budget configuration."},
 "grid": 10,
 "metric": "reported train/loss; Grug objective/version pending",
 "prefix": [{"value": 11.209885215759277, "x": 0.02}, {"value": 10.238005956013998, "x": 0.04},
            {"value": 9.437644867669968, "x": 0.06},  {"value": 8.730281375703358, "x": 0.08},
            {"value": 8.179786023639497, "x": 0.1},   {"value": 7.7530267125084285, "x": 0.12},
            {"value": 7.474150050770152, "x": 0.14},  {"value": 7.189137186322894, "x": 0.16},
            {"value": 6.961845102764311, "x": 0.18},  {"value": 6.787371090480259, "x": 0.2}],
 "query_x": [0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
 "task_semantics": "Predict or assign the recorded metric; source-specific objective terms may
    differ. Missing context is unknown."}

<CODE>
def loss_fn(params):
    compute_params = mp.cast_to_compute(params)
    return compute_params.next_token_loss(
        batch.tokens,
        batch.loss_weight,
        mask=batch.attn_mask,
        reduction="mean",
        logsumexp_weight=z_loss,
        return_router_metrics=True,
    )

(loss, summarized_metrics), grads = jax.value_and_grad(loss_fn, has_aux=True)(qb_params)
metrics = {"train/loss": loss, **summarized_metrics}
updates, opt_state = optimizer.update(grads, state.opt_state, qb_params)
params = optax.apply_updates(qb_params, updates)
```

The answer of Claude Opus 5 (999 output tokens; the parsed JSON object, with non-ASCII arrows and signs written in ASCII):

```
{"predictions": [{"x": 0.3, "p10": 5.85, "p50": 6.2,  "p90": 6.55},
                 {"x": 0.4, "p10": 5.5,  "p50": 5.9,  "p90": 6.3},
                 {"x": 0.5, "p10": 5.28, "p50": 5.7,  "p90": 6.12},
                 {"x": 0.6, "p10": 5.1,  "p50": 5.54, "p90": 5.98},
                 {"x": 0.7, "p10": 4.95, "p50": 5.4,  "p90": 5.85},
                 {"x": 0.8, "p10": 4.8,  "p50": 5.26, "p90": 5.75},
                 {"x": 0.9, "p10": 4.65, "p50": 5.12, "p90": 5.62},
                 {"x": 1.0, "p10": 4.48, "p50": 4.98, "p90": 5.5}],
 "rationale": {
```

Table 31: Forecasts at the 10th, 50th, and 90th percentiles of the error ratio R in each cell for Claude Opus 5 under CURVE, GRID10. R is the run's MAE over that of repeating the last value (lower is better); the quotation is one verbatim rationale item.

Cell	Pick	R	Rationale (verbatim)
FT / HF	good	0.20	"Log/power fit on last points implies ∼0.39–0.45 by x=1.0"
FT / HF	median	0.49	"Curve shape is convex decay approaching a plateau near ∼0.66–0.68"
FT / HF	bad	0.89	"A step-like drop between x=0.10 and 0.12 suggests a schedule change"
FT / Marin	good	0.08	"Decrement per 0.02 step shrinks from ∼0.17 to ∼0.011"
FT / Marin	median	0.15	"Log fit on last points gives slope ≈ -0.08 per ln-unit of x"
FT / Marin	bad	0.33	"Curve already near-plateau by x=0.2, typical of SFT loss after warmup"
PT / AI2	good	0.07	"∼0.50 loss drop from x=0.05→0.10, ∼0.26 from 0.10→0.20"
PT / AI2	median	0.63	"Reported metric stays the same cross-entropy scale (unusually high plateau retained)"
PT / AI2	bad	1.51	"Final loss plateaus around 2.5-3.4 for this model/data scale"
PT / Marin	good	0.07	"Value at x=0.2 is 3.872, suggesting slow log-like decay thereafter"
PT / Marin	median	0.24	"No plateau reached by x=0.2, so continued slow improvement expected"
PT / Marin	bad	0.60	"Plateau value ∼2.30 is near ln(10), suggesting a near-degenerate/uniform-prediction regime"
PT / MLF	good	0.18	"Offset power-law fit on x=0.1/0.14/0.2 gives L≈2.48+0.26·x$^{-0.59}$"
PT / MLF	median	0.32	"last points nearly linear-in-log-x with slowly shrinking increments"
PT / MLF	bad	0.81	"asymptotic loss level (2.1 vs 3.3) depends on unknown model/data scale and schedule"


```
"evidence": [
  "Prefix falls 11.21->6.79 over x=0.02->0.20 with rapidly shrinking increments (1.39 over
   last doubling).",
  "Very small model (num_layers=1, hidden_dim=512, 8 heads) limits attainable loss despite
   8192 seq len and batch 64.",
  "z_loss_weight=1e-4 plus possible router aux terms add a small positive offset to
   reported train/loss."],
"assumptions": [
  "Loss continues an approximately power-law/log decay with ~0.5-0.6 nats per doubling of x,
   flattening late.",
  "Cosine-style LR decay near end of the 1058-step budget yields a modest extra drop after
   x~0.8.",
  "No divergence, restarts, or mixture change in the remaining run."],
"uncertainty": "Moderate: shape of the tail is extrapolated from only 20% of the run; final
   value uncertain by roughly +/-0.5 nats, with asymmetric downside if late LR decay is
   strong."}}
```


The recorded values at the eight query points are 6.20, 5.89, 5.70, 5.50, 5.42, 5.28, 5.22, and 5.14. The median forecast has MAE 0.045 against 1.24 for repeating the last value ($R = 0.036$, $S = 0.96$), and all eight targets fall inside the 80% intervals; the same model given the curve alone (CURVE) has MAE 0.58 ($R = 0.47$) on this run.

Three forecasts per cell. Table 31 lists, for each pretraining and fine-tuning cell, the Claude Opus 5 CURVE (GRID10) forecast at the 10th, 50th, and 90th percentile of the per-run error ratio R within the cell, with one verbatim item of the model's stated rationale. The good forecasts fit an explicit power or log law to the shrinking decrements; the bad ones read a plateau or a schedule artifact into the prefix and hold it (fine-tuning on HF and Marin) or extrapolate a constant drop per doubling far past where it holds (AI2, MLFoundations).

Matching rationales under the swap. Every matching pair is asked twice, with curves 1 and 2 exchanged and settings A and B fixed. On the 300 pretraining and fine-tuning matching pairs of the test release (GRID10, each in two conditions), Claude Opus 5 answered both views validly for 419 of the 600 pair–condition combinations; 273 of these return probabilities that sum to one within 0.1 across the swap and 26 disagree with themselves by more than 0.5. The two pairs below are consistent: the rationale relabels the curves and the probability flips, once correctly and once wrongly.

- *Experts 256 vs. 64 (*FULL*, PT/Marin; correct in both views).* View `87a58266`: $P(\text{A=curve 1}) = 0.72$, "Curve 1 is uniformly lower than curve 2 from x=0.04 to 1.0". Swapped view `1078417c`: $P = 0.28$, "Curve 2 is uniformly lower than curve 1 from x=0.04 onward". Both views open with "Only differing field: model.num_experts 256 (A) vs 64 (B)" and assign the lower curve to the larger expert count.
- *Learning rate 0.0196 vs. 0.0037 (*FULL*, PT/Marin; wrong in both views).* View `ef31167a`: $P = 0.22$, "Curve 2's first logged point is much higher (9.01 vs 7.63)", attributed to the larger step size. Swapped view `911a0348`: $P = 0.78$, "Curve 1 starts

much higher (9.01 vs 7.63) at x=0.02, then drops faster". The story flips with the labels and is stable, but the run that starts higher and plateaus early is the *smaller*-rate run, so the model is confidently wrong twice.

One agent transcript. One attempt from the pilot, in which the agent + search arm also receives the curves of related runs (arm 4 of Table 30; Claude Opus 5, MLFoundations pretraining, 8 assistant turns, 172 s, $0.61); in the agent study of §4.5 the folder holds no related runs. Tool results are summarized in brackets, and the model's hidden reasoning blocks are omitted.

```
USER  Your task folder is the current directory. Start with `list_files` and
      `read_file README.md`. Finish by writing `answer.json` with `write_file`,
      then reply DONE.
AGENT list_files; read_file README.md
      [19 files: README, prefix.csv (10 rows), config.json, context.md,
       experiment_log.md, codebase/open_lm/{data,main,scheduler,train}.py,
       related_runs/related_0{1,2,3}/{config.json,curve.csv}, tools/fit_helpers.py]
AGENT read_file context.md, config.json, prefix.csv, experiment_log.md,
      related_runs/README.md
      [context: LM pretraining, MLFoundations scaling collection; loss reduction
       and auxiliary terms to be checked in source. config: planned_tokens 3.48e9,
       batch 32 x accum 4, warmup 2000, cosine to min lr. prefix: 9.63 -> 4.74
       over x = 0.02..0.20]
AGENT run_python: print the three related configs and curves
      read_file codebase/open_lm/scheduler.py
      [related runs: 1.4B params / 28.8B tokens, final loss 2.48-2.74;
       scheduler: linear warmup then cosine decay to a floor]
USER  You have not written a valid `answer.json` yet. Write it now with
      `write_file` ..., then reply DONE.
AGENT paper_search "loss curve prediction cosine schedule multi-power law
      learning rate decay" (limit 4)
      code_search "open_lm_160m" (limit 5)
      [paper_search: 4 hits, first is 2503.12811 "A Multi-Power Law for Loss Curve
       Prediction Across Learning Rate Schedules"; code_search: no results,
       project-internal hits are filtered out]
AGENT run_python: scipy curve_fit of L = L_inf + A x^-a on the prefix (x >= 0.06),
      plus an exponential; print values at x = 0.4, 0.6, 0.8, 1.0
      [power law: 3.93, 3.51, 3.22, 3.01; exponential: 4.36, ...]
AGENT write_file answer.json
      predictions: x=0.4 (3.63, 3.92, 4.32)  x=0.6 (3.28, 3.58, 4.02)
                   x=0.8 (3.08, 3.40, 3.84)  x=1.0 (2.98, 3.30, 3.76)
      evidence:
       - config/experiment_log: 6608 steps, warmup 2000 (30% of budget), so per
         scheduler.py the LR is still ramping at the end of the prefix and decays
         to the floor only at x=1.0; most of the remaining drop is still ahead.
       - related_runs: three 1.4B/28.8B runs finish at 2.48-2.74; scaling the
         20x-token form to ~1.7e8 params / 3.5e9 tokens gives ~3.2, the related
         runs sit 0.0-0.25 above it, so ~3.3 median with ~0.25 spread.
       - paper_search (Multi-Power Law) supports a front-loaded decrease plus an
         extra drop in the LR-decay phase; it moved ~55% of the remaining drop
         into 0.2->0.4 and kept a visible 0.8->1.0 drop, but did not change the level.
      assumptions: plain next-token cross-entropy in nats (z_loss 0); no
       divergence; cosine decay completes.
      uncertainty: unknown dataset/tokenizer level (+-0.15); the unusually long 30%
       warmup and high lr without qk_norm skew the upper tail; no post-warmup
       prefix points pin the decay shape.
      [wrote answer.json (1994 characters)]
AGENT DONE
```

J.2 PROMPTS

Prediction prompt. Shared by the three prediction conditions, which differ only in the `INPUT` object that follows it.

```
Forecast the recorded metric at every query_x from only the supplied prefix and
available context. No browsing, experiment identification, or remembered final
results. Return one JSON object: {"predictions":[{"x":number,"p10":number,"p50":numb
er,"p90":number}],"rationale":{"evidence":[short observations citing input
fields],"assumptions":[short assumptions],"uncertainty":short string}}. Cover
exactly query_x in order; use finite ordered quantiles in the original metric units.
Rationale must be concise, at most three evidence items and three assumptions.
Missing information is unknown. Do not provide a chain of thought.
```

Matching prompt. Shared by the two matching conditions.

```
Assign the two visible recorded-metric curves to the reported settings A and B using
only the supplied input. Curve order is arbitrary. This is assignment, not causal
identification; unobserved settings may differ. Return one JSON object:
{"probability_A_curve1":number,"rationale":{"evidence":[short observations citing
input fields],"assumptions":[short assumptions],"uncertainty":short string}}.
Probability must lie in [0,1]; the alternative assignment is its complement.
Rationale must be concise, at most three evidence items and three assumptions. No
```

```
browsing, experiment identification, or remembered final results. Do not provide a
chain of thought.
```

Prediction input, FULL (`enough_info_available`). The input of the default condition. The CURVE input (`curve_only`) omits the `context` object; the CURVE+CONFIG input (`curve_plus_config`) replaces it with a top-level `reported_configuration`, the run's flat reported configuration. Angle-bracket tokens are per-request values: the axis and metric descriptions are each one of five fixed strings, the task description is one of eight source-level paragraphs, and the interpretation rule is identical in all requests.

```
<PREDICTION_PROMPT>
INPUT:
{"axis": "<AXIS_DESCRIPTION>",
 "case": "<CASE_HASH>",
 "context": {
   "code_context": [{"binding": "<BINDING_LABEL>", "code": "<SOURCE_EXCERPT>",
                     "language": "python", "line_end": <int>, "line_start": <int>,
                     "source_sha256": "<SHA256>", "source_url": "<REPO_URL>"}, ...],
   "interpretation_rule": "Reported per-run configuration takes precedence over
        source example defaults. Project-level code is explanatory context, not
        proof of execution. Missing information is unknown, not zero.",
   "known_unknowns": ["<SENTENCE>", ...],
   "reported_configuration": {"<dotted.config.key>": <value>, ...},
   "source_facts": ["<SENTENCE>", ...],
   "task_description": "<ONE_PARAGRAPH_TASK_DESCRIPTION>"},
 "grid": <10 or 20>,
 "metric": "<METRIC_DESCRIPTION>",
 "prefix": [{"value": <v_1>, "x": <x_1>}, ..., {"value": <v_k>, "x": <x_k>}],
 "query_x": [<x_{k+1}>, ..., <x_grid>],
 "task_semantics": "Predict or assign the recorded metric; source-specific
                    objective terms may differ. Missing context is unknown."}
```

Matching input, FULL (`enough_info_available`). Two complete curves in arbitrary order, the two reported configurations, and one `context` object per setting with the same six fields as the prediction context. The CURVES+SETTINGS input (`curves_plus_minimal_settings`) omits the `contexts` object. The swapped view of a pair shares its `pair_id` and exchanges curves 1 and 2 while settings A and B stay fixed.

```
<MATCHING_PROMPT>
INPUT:
{"axis": "<AXIS_DESCRIPTION>",
 "case": "<CASE_HASH>",
 "contexts": {"A": {"code_context": [...], "interpretation_rule": "<AS ABOVE>",
                    "known_unknowns": [...], "reported_configuration": {...},
                    "source_facts": [...], "task_description": "<...>"},
              "B": {<same six fields for setting B>}},
 "curves": {"1": [{"value": <v>, "x": <x>}, ...], "2": [{"value": <v>, "x": <x>}, ...]},
 "grid": <10 or 20>,
 "metric": "<METRIC_DESCRIPTION>",
 "pair_id": "<PAIR_ID>",
 "settings": {"A": {"<dotted.config.key>": <value>, ...},
              "B": {"<dotted.config.key>": <value>, ...}},
 "statistic": "inherited observed trailing-window means; no new interpolation",
 "task_semantics": "Predict or assign the recorded metric; source-specific
                    objective terms may differ. Missing context is unknown."}
```

Agent folder `README.md`, agent study. The file an agent reads first in both arms of the agent study (§4.5); the axis and metric strings and the query coordinates vary per task, and everything else is identical in all 1,422 README files returned to the agents (checked by diffing); long lines are wrapped for display.

```
# Forecast this training metric

Read prefix.csv, config.json, context.md, experiment_log.md, and codebase/ as useful. There
are NO related runs. The log is config-derived. Source code is limited to the frozen evidence;
missing code is unknown. Do not identify this run or seek its final results. Offline unless
explicit search tools are provided.

{
  "axis": "<AXIS_DESCRIPTION>",
  "metric": "<METRIC_DESCRIPTION>"
}

Write answer.json using this schema, finite ordered quantiles in original units, exactly the
query coordinates:
{"predictions": [{"x": <x_{k+1}>, "p10": "number", "p50": "number", "p90": "number"}, ...,
 {"x": <x_grid>, "p10": "number", "p50": "number", "p90": "number"}], "rationale":
 {"evidence": [], "assumptions": [], "uncertainty": "short string"}}
```

Agent folder `README.md`, pilot. The file an agent read first in the pilot (arms 3, 4, and 4′ of Table 30). Reproduced from the task folder of one arm-4 run; the metric line, the axis line, the query points, the prefix row count, and the number of related runs (2–5) vary per task, and two sources add a short *Note on `codebase/`* paragraph. Everything else is identical across the 100 arm-4 runs (checked by diffing the README returned to each agent). The *Offline* rule is overridden for the agent + search arm by the system prompt below.

```
# Forecasting task

You are given the first 20% of one machine-learning training run's logged metric
curve, the configuration that was reported for that run, a launch-style experiment
log, the project's relevant source code, and a few complete curves from **other**
runs of the same project. Predict the metric at the remaining query points.
## What to predict
Metric: <METRIC_DESCRIPTION>
X axis: <AXIS_DESCRIPTION>
Query points (x): <x_{k+1}, ..., x_grid>

For every query x give three ordered quantiles of the metric in its **original
units**: `p10` (10th percentile), `p50` (median), `p90` (90th percentile).
## Output
Write a single file named **`answer.json`** in this folder, containing exactly one
JSON object:  {"predictions": [{"x": <x>, "p10": 0.0, "p50": 0.0, "p90": 0.0}, ...],
  "rationale": {"evidence": ["short observations citing input fields or files you
  read"], "assumptions": ["short assumptions"], "uncertainty": "one short string"}}
Cover exactly the query x above, in order, with finite values and `p10 <= p50 <= p90`.
At most three evidence items and three assumptions.
## Rules
- **Offline.** No network access, no browsing, no package installs beyond what is
  already present. Everything you need is in this folder.
- **Do not try to identify the run.** Do not search for, guess, or rely on remembered
  final results for a specific public experiment. Predict from the supplied prefix,
  configuration, code and related runs only.
- You may write and run Python. `numpy`, `scipy` and `pandas` are expected to be
  available (see `requirements.txt`); `tools/fit_helpers.py` gives you a starting point.
- Missing information is unknown, not zero.
- Suggested time budget: **15 minutes**. Optional: note what you did in `agent_trace.md`.
## Files in this folder
README.md, prefix.csv (x, value; <k> rows), config.json (reported configuration),
context.md (task description, source facts, interpretation rule, known unknowns,
source excerpts), experiment_log.md (launch-style summary derived from the config),
codebase/ (real source files at the evidenced commit, with MANIFEST.json),
related_runs/ (<n> complete curves from **other** runs of the same project, each
with its own config.json), tools/fit_helpers.py, requirements.txt
```

Agent + search system prompt addendum. Rules prepended for the agent + search arm, in the pilot and in the agent study (where the sentence "There are no related runs." is appended); they override the README's offline rule. Verbatim from the runner (`TOOL_RULES`).

```
You have a small number of tools. Rules that override the folder's README:
- The README says "Offline. No network access". For THIS run that rule is replaced by:
  you have exactly two outward-facing tools, `paper_search`/`paper_fetch` and
  `code_search`. There is no general web browsing, no URL fetching, and the Python
  sandbox has no network.
- You may NOT try to find this run's logs, its Weights & Biases page, its final loss,
  or any published number for this specific experiment. Search for *general*
  knowledge: scaling-law and loss-curve forms, learning-rate schedule behaviour,
  optimizer/framework implementation details, typical loss levels for an
  architecture and token budget. Search results that come from this project's own
  repositories are filtered out before you see them.
- Do not try to identify which public run this is.
- **Use the search tools.** Before you finalise the answer you must issue at least
  one `paper_search` and at least one `code_search` whose results could bear on this
  forecast, and you must say in `rationale.evidence` whether what you found changed
  your prediction. If the results are useless, say so and ignore them -- but look first.
- Everything else in the folder's README still holds. Write your final `answer.json`
  into the task folder with `write_file` before you finish.
```