CAN LLM AGENTS ESTIMATE THEIR OWN TASK-COMPLETION TIME HORIZON?

Anonymous authors
Paper under double-blind review

ABSTRACT

The increasing reliability of LLM agents has led to soaring demand for persistent assistants and autonomously running agents. Yet an overlooked limitation of current systems is their inability to reliably model how long their own runs will take. Despite its practical importance for agent scheduling, planning, and real-time coordination, this phenomenon has not been systematically studied. To bridge this gap, we propose an evaluation framework that decomposes run-time self-estimation into four categories: pre-run, post-run, relative, and compositional estimation. We further investigate prompting- and training-based calibration methods to reduce estimation error. We find that models appear to inherit general knowledge about task duration from training data, yet they anticipate poorly how much work their own runs will involve and how the deployment that executes them affects run time. These results expose an important weakness in how modern LLMs anticipate their own run time, with practical implications for persistent agents, time-sensitive applications, and real-time human–AI coordination.

1 INTRODUCTION

As LLM agents increasingly tackle longer-horizon tasks in engineering and AI R&D (Kwa et al., 2025; Wijk et al., 2025), they are also being envisioned as personalized, always-on assistants that maintain context and act proactively over extended periods (Lu et al., 2025; Zhang et al., 2026; Lin et al., 2026; Chen & Chen, 2026). In these settings, agents must reason not only about what actions to take, but also about how long those actions are likely to take. Unreliable estimates can distort human expectations, disrupt scheduling and coordination, and lead to poor resource allocation. These errors can become particularly consequential in nested workflows and multi-agent systems, where inaccuracies may compound across successive stages.

The stakes are higher when timing itself determines success (Xie et al., 2024; Xu et al., 2024). In domains such as medical triage and emergency response, misjudging duration can have direct operational consequences (Sravanthi et al., 2023; Shi et al., 2025). If LLMs are to operate over extended periods, they need *at least* coarse temporal calibration about their own behaviour.

Runtime estimation is further complicated by deployment, as the same model can exhibit different latencies across inference modes, hardware, and system configurations. This deployment sensitivity further decouples semantic task difficulty from observed duration, so an estimate must combine a forecast of the work a task needs with knowledge of how fast the deployment performs that work. Yet despite its importance, the ability of LLM agents to estimate their own run time has not been systematically characterized.

To study this systematically, we formulate run-time self-estimation as a four-part evaluation problem. The framework isolates four capabilities: prospective estimation before acting, comparative estimation over pairs of tasks, retrospective estimation after a run ends, and compositional estimation across chained subtasks. This decomposition separates failures of scale, ranking, recall and error propagation.

Our work makes five key contributions:

- **Evaluation Framework:** We introduce MetaTime, a benchmark for evaluating run-time self-estimation in LLM agents across four complementary settings.

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.

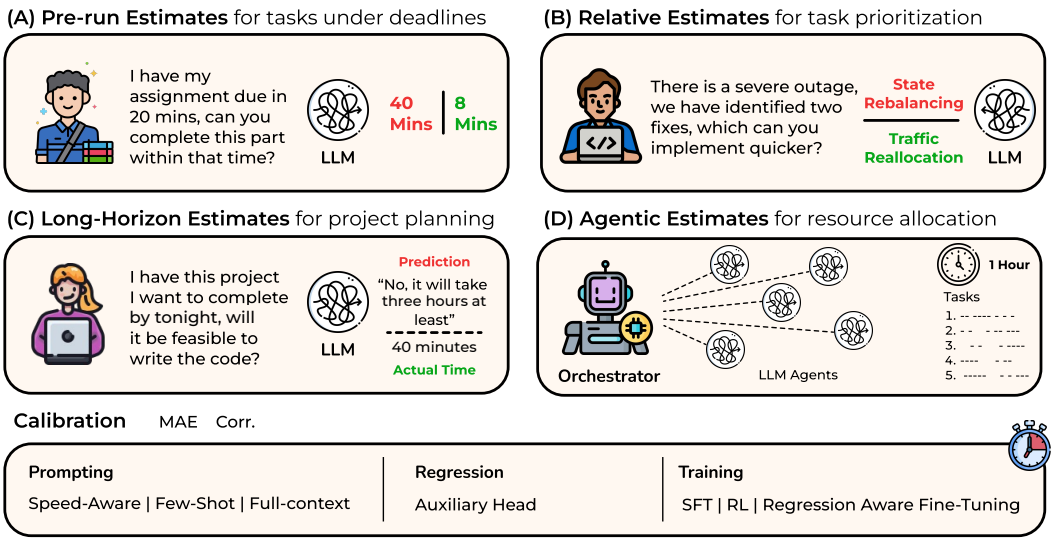

Figure 1: Overview of the evaluation framework. We decompose run-time self-estimation into four settings and evaluate them across long-horizon agentic, short agentic and single-turn tasks.

- **Empirical Results:** Across tasks spanning seconds to nearly two hours, we find that run-time self-estimation is broadly and consistently poor across all evaluation settings.
- **Calibration:** We find that simple prompting, and training on an open model, can partially reduce estimation error, but no method fully closes the gap.
- **Deployment Sensitivity:** We provide measures to better quantify deployment sensitivity and highlight the need for stronger deployment-aware run-time calibration.
- **Estimates Under Incentive:** Lastly, we show that when a model is told its estimate will become its time limit, it inflates the estimate, and its reasoning usually does not say why, while agents asked to take more or less time barely change how long they run.

2 Problem Formulation

Given a task t, model M, and deployment configuration d, we define run-time self-estimation as the model's estimate $\hat{y}$ of its own wall-clock execution time y. The estimate is conditioned on the context available when it is elicited, while the observed duration may additionally depend on deployment factors such as inference speed, hardware, and tool latencies. A run's duration therefore has two parts: the amount of work the task requires, in model calls, tool calls and generated tokens, and the rate at which the deployment turns that work into time. A useful estimate needs both, and our experiments probe each of them. We study four complementary settings, illustrated in Figure 1.

Prospective estimation asks the model how long it expects to take before beginning a task, whereas retrospective estimation asks it to estimate the duration once the run has ended, from the record of that run. Comparative estimation tests whether the model can correctly compare the expected durations of different tasks, while compositional estimation tests whether duration estimates remain calibrated when reasoning across chained subtasks. Together, these settings probe absolute scale, relative ordering, inference from completed work, and error propagation.

The main evaluation in Sections 3–5 covers all four settings at agent scope: prospective and retrospective estimation on every task, comparative estimation over task pairs, and compositional estimation over chained tasks. Calibration by training, which requires access to model weights, is studied on an open model in Appendix C.

Run-time boundary. The target duration y includes model inference, tool execution, and waiting within the run, while environment provisioning before the agent starts and grading after it terminates are excluded, and the scored duration is the time until the attempt stops, whether or not it succeeds;

stating this stopping point in the prospective question does not significantly change accuracy (Appendix B.11). We keep this boundary fixed across settings and state it explicitly in the estimation prompt so that the model answers the same question we score.

Metrics. We evaluate two complementary aspects of run-time estimation. Mean absolute error (MAE) measures absolute deviation in wall-clock seconds, while Pearson correlation measures how strongly estimated durations track variation in observed durations:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|, \qquad r = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{\hat{y}})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(\hat{y}_i - \bar{\hat{y}})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}.$$

At agent scope, observed durations span approximately two orders of magnitude, so we compute r on log-transformed predicted and observed durations. We also report the median ratio $\hat{y}/y$, which captures the direction and magnitude of systematic over- or under-estimation on a scale that remains comparable across task horizons, and the median absolute log error $|\log(\hat{y}/y)|$, where 0.69 is a factor of two in either direction.

3 Dataset

We introduce MetaTime, a benchmark for run-time self-estimation spanning 182 tasks across three time scales. The benchmark combines 95 long-horizon agentic tasks whose runs typically last tens of minutes, 19 shorter agentic tasks lasting seconds to minutes, and 68 single-turn tasks lasting seconds. This lets us study the same self-estimation problem as the duration of the underlying work grows by several orders of magnitude.

Regime	Source	Tasks	Median
Long-horizon	SkillsBench	34	8–24 min
	Harbor-Index	11	19–58 min
	DeepSWE	38	11–19 min
	LHTB	6	11–40 min
	Toolathlon	6	10–24 min
Short agentic	Custom	11	34–48 s
	METR public	8	62–75 s
Single-turn	7 categories	68	12–15 s
Total		**182**	

Table 1: Composition of MetaTime. Duration gives the range of per-model medians across GPT-5.5 and Claude Opus 4.8.

3.1 Long-horizon agentic tasks

The main evaluation contains 95 tasks drawn from five agent benchmarks: SkillsBench (Li et al., 2026b) (34), Harbor-Index (Shi et al., 2026) (11), DeepSWE (Huang et al., 2026) (38), the Long-Horizon Terminal Bench (LHTB) (Li et al., 2026c) (6), and Toolathlon-Verified (Li et al., 2026a) (6). These benchmarks cover substantially different forms of agentic work and interaction, including software engineering, scientific analysis, planning and control, professional analytical work, document and media creation, and cross-application tool use.

Software and data-system engineering account for 53 of the 95 tasks, while the remaining 42 extend the set to scientific and ML workflows, planning and optimization, legal and consulting work, document and media manipulation, and business operations. Examples include seismic phase picking, exoplanet period detection, drone planning, flood-risk analysis, court-form filling, and video processing. Appendix B gives the complete task list and source breakdown.

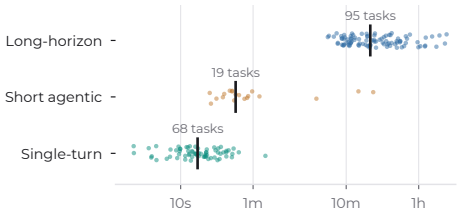

Figure 2: Recorded durations of the runs behind each regime, on a log scale, with the median marked. The benchmark spans roughly three orders of magnitude in wall-clock time.

For each task, we use its recorded benchmark execution to obtain its wall-clock duration and the model-visible context available before and after execution. We retain tasks with non-trivial run times and sufficiently complete execution records, while preserving both successful and unsuccessful runs. Detailed inclusion criteria and benchmark-specific execution

details are given in Appendix B. Across the resulting set, individual runs range from a few minutes to nearly two hours.

3.2 Short agentic tasks

The short agentic set contains 19 tasks executed with the models' native coding agents. Eleven custom tasks cover log analysis, scheduling, code repair, incident response, data reconciliation, and small application development, while eight tasks from the METR public evaluation suite (Kwa et al., 2025) cover debugging, symbolic regression, scientific inference, software engineering, and game implementation. These tasks provide an intermediate regime between single responses and the longer benchmark runs.

3.3 Single-turn tasks

The single-turn set contains 68 tasks spanning code generation, reasoning, summarization, writing, debugging, creative writing, and question answering. Each task consists of a single prompt answered without tools, with execution times ranging from a few seconds to roughly a minute. This set additionally supports the compositional experiments in Section 5.3 and the calibration-by-training experiments in Appendix C.

4 Experimental Setup

Models. We evaluate GPT-5.5 and Claude Opus 4.8 at medium reasoning effort. Each model estimates the duration of its own runs, so the executing and estimating model are always matched. Calibration by training requires model weights, so it uses the open Qwen3-8B (Appendix C).

Estimation protocol. For each task, we ask for a prospective estimate before execution and a retrospective estimate after execution. The prospective question is conditioned on the task context available when the agent begins, while the retrospective question is conditioned on the model-visible context at the end of the run. Both ask for the wall-clock duration defined in Section 2. For long-horizon tasks, these contexts are reconstructed from the released execution records; for the short agentic and single-turn tasks, we query the model directly within the corresponding execution setup. Information unavailable to the model during execution, including external timing metadata, is excluded. Exact prompts and reconstruction details appear in Appendix B.

Execution and evaluation. The short agentic tasks are executed with Claude Code for Opus 4.8 and the Codex CLI for GPT-5.5, while single-turn tasks use the same models with tools disabled. We report the median ratio between estimated and observed duration, the median absolute log error and the correlation on log duration as defined in Section 2, with mean absolute error in seconds in the appendix tables.

5 Results

5.1 Long tasks are hard to forecast and easier to judge afterwards

Before a long-horizon run starts, the models can barely tell which tasks will take longer. Their prospective estimates correlate with the actual durations at only 0.26 for GPT-5.5 and 0.07 for Opus 4.8 (Table 2). Figure 3a shows part of the reason: across 95 very different tasks, each model uses only about twenty distinct answers, and most of them are round values such as 25 or 45 minutes. This pattern is consistent with the models placing a task in a rough size class and picking a familiar number for it, and it is why their estimates form horizontal bands in Figure 4. What little ordering signal they have lies within each benchmark, since removing each benchmark's mean duration raises Opus 4.8's correlation to 0.30 (Appendix B.12). Much of what can be known about a long run's duration is the setting it runs in: predicting each run by the average duration of the other runs in its benchmark correlates with the actual durations at 0.50, whereas the other model's run of the same task correlates at only 0.15.

		Prospective			Retrospective		
Regime	**Model**	**Ratio** $\to 1\times$	**Log err.** $\downarrow$	**Corr.** $r \uparrow$	**Ratio** $\to 1\times$	**Log err.** $\downarrow$	**Corr.** $r \uparrow$
Long-horizon	GPT-5.5	2.39×	1.00	0.26 [.05, .48]	1.07×	0.27	0.76 [.68, .84]
	Opus 4.8	1.01×	0.81	0.07 [−.16, .26]	0.76×	0.40	0.64 [.54, .73]
Short agentic	GPT-5.5	7.96×	2.07	0.70 [.02, .87]	0.45×	0.99	0.67 [.27, .87]
	Opus 4.8	3.99×	1.38	0.87 [.24, .95]	1.26×	0.43	0.92 [.61, .97]
Single-turn	GPT-5.5	3.47×	1.24	0.55 [.33, .73]	0.93×	0.30	0.67 [.53, .79]
	Opus 4.8	2.87×	1.06	0.81 [.71, .87]	1.80×	0.59	0.81 [.69, .88]

Table 2: Run-time self-estimation across the three evaluation regimes. Ratio is the median estimate over the observed duration, so below $1\times$ underestimates and above overestimates. Log error is the median absolute error in log units, where 0.69 is a factor of two. r is the Pearson correlation on log durations. GPT-5.5's long-horizon figures use the 93 tasks with both answers. Darker cells are further from a correct answer.

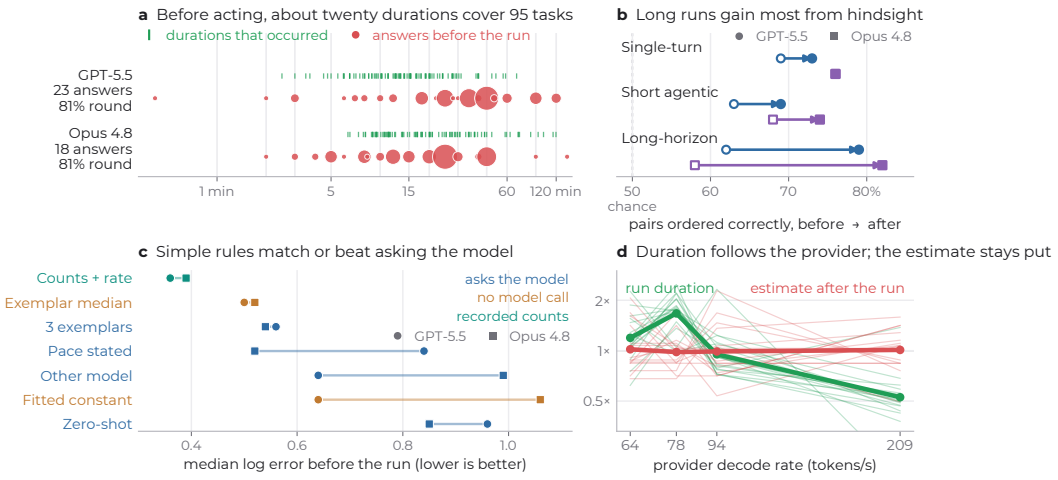

Figure 3: **(a)** Before the run, about twenty distinct durations cover 95 long-horizon tasks, and most sit on round values (faint guides). **(b)** Task pairs ordered correctly before and after the run. **(c)** Prospective error per intervention, with the label colour giving its kind. **(d)** One open model on four providers: each thin line is a task relative to its own mean, and the bold lines average them. The duration changes with the provider while the estimate stays flat.

After the run, the same models do much better, with correlations of 0.76 and 0.64 and median ratios close to one. At this point the model reads the transcript of the finished run, so it can see how much work was actually done, and Section 6 shows that this is mostly what it relies on.

On shorter tasks the main problem is scale. Before running them, the models overestimate short agentic and single-turn tasks by a factor of three to eight, and their estimates after the run are much closer. Part of the short agentic ordering reflects the gap between its two task families, since within the custom tasks the correlations vanish (Table 15). For reference, repeat runs of the same task on the same provider in our deployment study imply that even a perfect forecast of a task's typical duration would miss a single run by a median factor of about 1.3. Comparing pairs of tasks tells the same story (Table 3 and Figure 3b). Before a long run, the models order only about 60% of task pairs correctly,

	Before	**After**	**Direct**
Long-horizon GPT-5.5	62%	79%	57%
Long-horizon Opus 4.8	58%	82%	50%
Short agentic GPT-5.5	63%	69%	–
Short agentic Opus 4.8	68%	74%	–
Single-turn GPT-5.5	69%	73%	–
Single-turn Opus 4.8	76%	76%	–

Table 3: Task pairs ordered correctly, using the estimates before and after the run, and when asking directly which task takes longer. Chance is 50%.

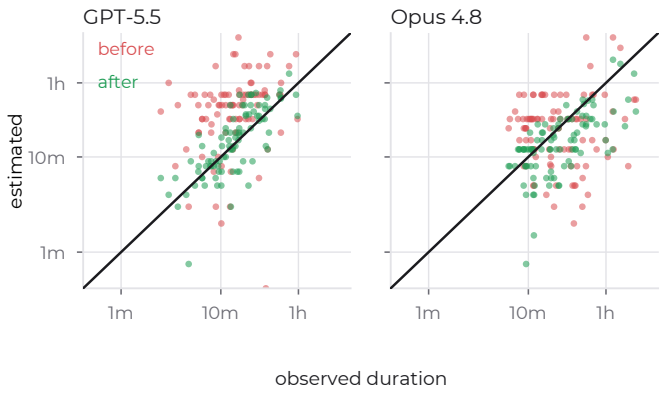

Figure 4: Estimated against observed duration on the long-horizon set, with the diagonal marking a correct answer. Before the run the estimates form horizontal bands, because a handful of round durations covers most tasks; afterwards they move toward the diagonal.

How far duration moves with the deployment	**Value**	**95%**
Elasticity of duration to decode rate	-0.85	$[-0.99, -0.68]$
Deployment shift in duration, SD in log units	0.49	
Deployment shift in message count, SD	0.14	
Deployment shift in tool-call count, SD	0.19	
Repeat noise in duration, SD in log units	0.39	
How far the estimates move with it	**Prospective**	**Retrospective**
Estimate shift across providers, SD in log units	0.15	0.22
Correlation with the actual shift	0.14	0.25

Table 4: Deployment sensitivity of GLM-5.3 on the 19 short agentic tasks served by four providers. The top half shows how much the duration moves with the provider, and the bottom half how much the estimates move with it. The elasticity is the slope of log duration on log decode rate after removing each task's mean, with a task-bootstrap interval.

where guessing gives 50%, and after the run this rises
to about 80%. Asking the model directly which of two tasks will take longer does no better than comparing its two separate estimates, so both questions draw on the same weak forecast.

5.2 Run time depends on the deployment

The same model can run at very different speeds depending on where it is served. To measure how much this matters, we run one open model, GLM-5.3, on the 19 short agentic tasks through four providers that serve identical weights at speeds between 64 and 209 tokens per second. The fastest provider finishes the same tasks about three times sooner than the slowest, even though the agent takes the same steps. The model's estimates hardly change across providers (Figure 3d), so the same answer is about ten times too long on one provider and three times too long on another. Telling the model the measured speed of its provider helps, yet it helps no more than a simple rule that rescales the model's answer by that speed (Table 12).

5.3 Errors grow when tasks are chained

Agents usually work through several tasks in a row, so we also group three single-turn tasks into one bundle and ask for an estimate of the whole bundle (Table 5). Running tasks together is faster than running them apart, since the recorded durations of the parts already add up to 1.5 to 1.8 times the bundle. Summing the models' estimates for the parts overshoots by four to seven times, which is mostly each task's overestimate carried into the sum. The single estimate for the whole bundle is where composition breaks down: it overshoots by seven to eighteen times, and on 12 bundles of five

Estimate of the bundle	GPT-5.5 Ratio	GPT-5.5 Log err.	GPT-5.5 Within 2×	Opus 4.8 Ratio	Opus 4.8 Log err.	Opus 4.8 Within 2×
Sum of the parts' recorded durations	1.81×	0.59	68%	1.51×	0.41	91%
Sum of the parts' own estimates	7.27×	1.98	0%	3.99×	1.38	0%
One estimate for the bundle, before	17.63×	2.87	0%	7.40×	2.00	0%
One estimate for the bundle, after	1.37×	0.34	86%	1.41×	0.36	91%

Table 5: Estimating a bundle of three single-turn tasks, scored against the bundle's actual duration. Each row is a different way to arrive at the estimate. Within 2× is the share of bundles estimated within a factor of two.

tasks it stays where it was, at 17.4 and 7.3 times, although the bundle now holds two thirds more work. This suggests the model answers with a typical value for a bundle of tasks. We see a similar pattern when we ask the open model partway through its runs how much time is left: it overshoots by about four times, and simply answering with the time already spent does better (Table 11).

5.4 Estimates change with how they will be used

An agent's time estimate is often used to decide its time limit or whether its task runs at all, so we ask whether the estimate changes once the model knows how it will be used. We repeat the prospective question on the 48 ablation tasks with one added sentence. In the *deadline* version, the estimate becomes a hard time limit and the attempt counts as failed if it runs over, which gives the model a reason to answer high. In the *approval* version, short estimates run first and long ones may be cancelled, which gives it a reason to answer low. Neither sentence changes the task, so a purely predictive estimate should stay the same. We ask each version with reasoning turned on and off; since Opus 4.8's thinking is hidden from us, we ask it to write out its reasoning before answering.

A deadline raises the estimates, whereas the approval rule leaves them unchanged (Table 6, Figure 5). GPT-5.5 nearly doubles its estimate under a deadline when it reasons, and barely changes it when it answers directly; Opus 4.8 raises it slightly once it writes out its reasoning. Adding a margin before a hard deadline is sensible behaviour, yet it means the number an operator receives depends on what the number will be used for. The reasoning seldom explains the change: GPT-5.5 mentions the deadline in fewer than half of the cases and Opus 4.8 almost never does, so someone reading the reasoning would usually not see why the estimate moved.

	Deadline	Approval	Says why
GPT-5.5, reasoning	1.81×	1.06×	43%
GPT-5.5, no reasoning	1.05×	1.00×	–
Opus 4.8, reasoning	1.14×	1.00×	4%
Opus 4.8, no reasoning	1.00×	1.00×	–

Table 6: Change in the estimate when one sentence about its use is added, as a multiple of the answer to the plain question (median over tasks). Says why is the share of tasks whose estimate rose and whose reasoning mentions the deadline.

5.5 Agents barely control their own run time

An estimate is most useful when the agent can act on it, so we add one sentence to each of the 19 short agentic tasks asking the agent to take about half, or double, its usual time on that task. Control is weak (Table 7). Asked to halve their time, GLM-5.3 and Opus 4.8 speed up by a quarter to a third and GPT-5.5 not at all. Asked to double it, GLM-5.3 and Opus 4.8 again finish faster, as if any budget meant hurry, while GPT-5.5 slows down, in part by waiting with sleep commands. Success rates barely change.

6 Ablations

To understand where the estimates come from, we keep each executed run fixed and change only what the model sees when it answers. All ablations use a subset of 48 long-horizon tasks that covers the five benchmarks in proportion.

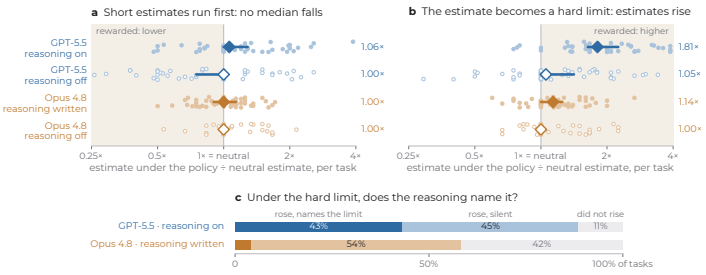

Figure 5: Each dot is one task: the estimate after one sentence about its use is added, divided by the answer to the plain question. Diamonds mark the median. **(a)** When short estimates are approved first, the estimates stay put. **(b)** When the estimate becomes a hard time limit, they rise. **(c)** Under the deadline, how often the estimate rose with the deadline mentioned in the reasoning, rose without it being mentioned, or did not rise.

	Asked to take	
Model	**Half**	**Double**
GLM-5.3	0.75×	0.73×
GPT-5.5	1.01×	1.74×
Opus 4.8	0.68×	0.73×
Target	*0.50*×	*2.00*×

Table 7: Run time when asked to take half or double the usual time, as a median multiple of the usual run on the short agentic tasks.

	GPT-5.5		Opus 4.8	
Model sees	**Ratio**	**Err.**	**Ratio**	**Err.**
Task only (before)	1.86×	0.87	0.76×	0.85
Full transcript	0.91×	0.52	0.61×	0.52
Task + work counts	1.22×	0.37	1.07×	0.43
Counts, fitted rate	0.94×	0.36	1.11×	0.39

Table 8: What the estimate after a run relies on. Work counts are the number of model requests, tool calls and output tokens; the last row converts them to seconds with a rate fitted on other tasks and uses no model. Err. is the median absolute log error.

The estimate after a run comes from the amount of work. The transcripts we show the model have execution-clock readings removed, and in the one block whose recorded contexts still contained them, removing them leaves the estimates unchanged. When we instead show the model only the task and three counts of the work that was done (model requests, tool calls and output tokens), it does better than with the full transcript, and a fixed conversion of these counts into seconds does just as well (Table 8). The model therefore, seemingly judges duration mainly by how much work it can see. As before a run it can partly predict this work, since Opus 4.8's predicted number of requests correlates with the real one at 0.76, yet both models expect about three times more output than they produce (Table 18). Since Table 8 covers only the tasks whose records include these counts, its first row differs slightly from the zero-shot row of Table 10.

Simple rules do as well as the model. Giving the model a sense of scale helps as three example durations from other tasks, the error before a run drops from about 0.9 to 0.55, yet simply answering the median of those three examples seems to do just as well (Figure 3, Table 10). Similarly, dividing the model's answers by one constant fitted on other tasks works about as well as telling it how fast it works. The models also take a stated speed at face value, and raise their estimates severalfold when told it is ten times slower. Finally, Opus 4.8 predicts the duration of GPT-5.5's runs better than GPT-5.5 does, and the two models' estimates for the same task correlate at 0.87, which suggests that both draw on a general sense of how long a task takes and know little that is specific to themselves. Raising the estimator's reasoning effort from medium to high makes no measurable difference (Table 13).

7 Related Work

Long-horizon and multi-step agents. Recent work studies increasingly long-horizon task execution in interactive environments, software engineering, real computer use, and broader agent benchmarks (Manakina et al., 2025; Jiang et al., 2026). Parallel work studies coordination and special-

ization in multi-agent and hierarchical systems (Tran et al., 2025; Ruan et al., 2026; Estornell et al., 2025). As these systems become more nested, scheduling, routing, and test-time compute allocation become part of the core problem (Gray et al., 2023; Erdogan et al., 2025; Paglieri et al., 2025). Our work focuses on a basic capability required in such settings: estimating how long the agent's own actions take.

Time in language models. Existing work examines temporal fact recall, event ordering, temporal validity across time, temporal point processes, and external time-series forecasting (Ding & Wang, 2025; Herel et al., 2024; Goel et al., 2025; Chen et al., 2026; Jin et al., 2024). In multimodal settings, recent benchmarks study spatio-temporal reasoning in video and report related failures of temporal sensitivity (Cheng et al., 2025; Song et al., 2025; Zhao et al., 2025; Upadhyay et al., 2025). Concurrent work also studies temporal blindness in agent tool-use decisions, asking whether agents act as if time has passed (Cheng et al., 2026).

Predicting length and confidence. Serving systems predict response length to schedule requests, budget-aware reasoning controls how many tokens a model spends (Han et al., 2025), and work on verbalized confidence calibrates what models say about their own accuracy. We ask the analogous question for run time.

Self-knowledge and monitoring. Work on introspection and situational awareness asks what models know about themselves and their circumstances (Binder et al., 2024; Lindsey, 2026; Berglund et al., 2023). Time horizons measure model capability by how long the same tasks take humans (Kwa et al., 2025), and recent work measures this horizon for models that answer without any visible reasoning (Gould et al., 2026). Our incentive experiment connects to this line of work, since we ask whether a model's time estimate changes with its use and whether its reasoning shows the change.

8 Conclusion

We study how well LLM agents can estimate their own task durations through a four-part evaluation framework and the MetaTime benchmark. Across single-turn, short agentic and long-horizon tasks, models show substantial errors in absolute duration, while their ability to anticipate which tasks will take longer is particularly weak at long horizons. Estimates improve after execution, when the model can reason from the work that occurred. Prompting and training can improve calibration, though substantial errors remain across the evaluated settings. The estimates also change with how they will be used, and the models' reasoning rarely says so, while agents asked to take more or less time barely change how long they run.

These findings suggest that better run-time self-estimation requires both anticipating the work a task will involve and accounting for the deployment in which it is executed. System clocks provide elapsed time, while models must use this information alongside estimates of the work ahead to make decisions under limited budgets. For example, an agent running experiments may need to decide whether to spend another hour on its current approach or move on to an alternative. We believe that learning to connect observed durations with future estimates is a useful direction for developing agents that can make such decisions, and MetaTime provides a basis for evaluating this capability.

Limitations. The deployment and incentive studies each cover a small set of tasks and models, and the incentive policies are described to the model without being enforced.

Reproducibility Statement

All experiments are reproducible from public artifacts. We release the task lists for the three regimes, the exact estimation prompts, the per-run records of context condition and reported usage, and the analysis code at `https://anonymous.4open.science/status/Time-Estimation-B286`. Long-horizon durations and contexts come from the released execution records of the source benchmarks, and Appendix B documents how they are reconstructed

and which strings are removed. For the short agentic and single-turn sets the full task suites, model configurations and hyperparameters are included in the supplementary material, with details in Appendix C; the deployment study pins model revision, quantization and provider for every run.

AI USE STATEMENT

LLM-based tools were used during development for data validation, plotting script generation, archive auditing and result consolidation. LLMs were also used for receiving feedback on writing structure and running experiments.

REFERENCES

Lukas Berglund, Asa Cooper Stickland, Mikita Balesni, Max Kaufmann, Meg Tong, Tomasz Korbak, Daniel Kokotajlo, and Owain Evans. Taken out of context: On measuring situational awareness in LLMs. *arXiv preprint arXiv:2309.00667*, 2023. URL https://arxiv.org/abs/2309.00667.

Felix J Binder, James Chua, Tomek Korbak, Henry Sleight, John Hughes, Robert Long, Ethan Perez, Miles Turpin, and Owain Evans. Looking inward: Language models can learn about themselves by introspection. *arXiv preprint arXiv:2410.13787*, 2024. URL https://arxiv.org/abs/2410.13787.

Chen Chen and Zhehuai Chen. Jarvisbench: Always-on intelligence between humans and agents, 2026. URL https://arxiv.org/abs/2608.14870.

Lili Chen, Wensheng Gan, Shuang Liang, and Philip S. Yu. Enhancing temporal awareness in LLMs for temporal point processes, 2026. URL https://arxiv.org/abs/2601.00845.

Yize Cheng, Arshia Soltani Moakhar, Chenrui Fan, Parsa Hosseini, Kazem Faghih, Zahra Sodagar, Wenxiao Wang, and Soheil Feizi. Your llm agents are temporally blind: The misalignment between tool use decisions and human time perception, 2026. URL https://arxiv.org/abs/2510.23853.

Zixu Cheng, Jian Hu, Ziquan Liu, Chenyang Si, Wei Li, and Shaogang Gong. V-STaR: Benchmarking video-llms on video spatio-temporal reasoning, 2025. URL https://arxiv.org/abs/2503.11495.

Xi Ding and Lei Wang. Do language models understand time? In *Proceedings of the ACM Web Conference 2025 Workshop on Telling Time in the Age of Large Language Models*, 2025. doi: 10.1145/3701716.3717744. URL https://doi.org/10.1145/3701716.3717744.

Lutfi Eren Erdogan, Hiroki Furuta, Sehoon Kim, Nicholas Lee, Suhong Moon, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. Plan-and-act: Improving planning of agents for long-horizon tasks. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=ybA4EcMmUZ.

Andrew Estornell, Jean-Francois Ton, Muhammad Faaiz Taufiq, and Hang Li. How to train a leader: Hierarchical reasoning in multi-agent LLMs, 2025. URL https://openreview.net/forum?id=SAtokoHpij. Submitted to ICLR 2026.

Krish Goel, Sanskar Pandey, KS Mahadevan, Harsh Kumar, and Vishesh Khadaria. Chronocept: Instilling a sense of time in machines, 2025. URL https://arxiv.org/abs/2505.07637.

Dewi Gould, Francis Rhys Ward, Anders Cairns Woodruff, Rauno Arike, Josh Hills, Alex Serrano, Ida Caspary, Jason Ross Brown, Jo J. Jiao, Patrick Leask, Twm Stone, Ram Potham, Ionut Gabriel Stan, Harry Mayne, Simeon Hellsten, Shubhorup Biswas, Ariana Azarbal, William L. Anderson, Elle Najt, Ryan Greenblatt, and Julian Stastny. Think fast: Estimating no-cot task-completion time horizons of frontier ai models, 2026. URL https://arxiv.org/abs/2606.07157.

Margaret Anne Gray, Zhuorui Yong, Abhijan Wasti, Esa M. Rantanen, and Jamison Heard. Measuring temporal awareness for human-aware ai. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 67:1817 – 1823, 2023. URL https://api.semanticscholar.org/CorpusID:264503405.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware llm reasoning, 2025. URL https://arxiv.org/abs/2412.18547.

David Herel, Vojtech Bartek, Jiri Jirak, and Tomas Mikolov. Time awareness in large language models: Benchmarking fact recall across time, 2024. URL https://arxiv.org/abs/2409.13338.

Wenqi Huang, Charley Lee, Leonard Tng, and Serena Ge. Deepswe: Measuring frontier coding agents on original, long-horizon engineering tasks, 2026. URL https://arxiv.org/abs/2607.07946.

Tanqiu Jiang, Yuhui Wang, Jiacheng Liang, and Ting Wang. Agentlab: Benchmarking llm agents against long-horizon attacks, 2026. URL https://arxiv.org/abs/2602.16901.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-LLM: Time series forecasting by reprogramming large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Unb5CVPtae.

Thomas Kwa, Ben West, Joel Becker, Amy Deng, Katharyn Garcia, Max Hasin, Sami Jawhar, Megan Kinniment, Nate Rush, Sydney Von Arx, Ryan Bloom, Thomas Broadley, Haoxing Du, Brian Goodrich, Nikola Jurkovic, Luke Harold Miles, Seraphina Nix, Tao Lin, Neev Parikh, David Rein, Lucas Jun Koba Sato, Hjalmar Wijk, Daniel M. Ziegler, Elizabeth Barnes, and Lawrence Chan. Measuring AI ability to complete long software tasks. *arXiv preprint arXiv:2503.14499*, 2025. URL https://arxiv.org/abs/2503.14499.

Junlong Li, Wenshuo Zhao, Jian Zhao, Weihao Zeng, Haoze Wu, Xiaochen Wang, Rui Ge, Yuxuan Cao, Yuzhen Huang, Wei Liu, Junteng Liu, Zhaochen Su, Yiyang Guo, Fan Zhou, Lueyang Zhang, Juan Michelini, Xingyao Wang, Xiang Yue, Shuyan Zhou, Graham Neubig, and Junxian He. The tool decathlon: Benchmarking language agents for diverse, realistic, and long-horizon task execution, 2026a. URL https://arxiv.org/abs/2510.25726.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, Chujun Tao, Binxu Li, Xuandong Zhao, Hejia Geng, Xiaojun Wu, Junwei Zhou, Xiaokun Chen, Hanwen Xing, Yubo Li, Qunhong Zeng, Di Wang, Yuanli Wang, Roey Ben Chaim, Penghao Jiang, Haotian Shen, Luyang Kong, Xinyi Liu, Runhui Wang, Xuanqing Liu, Jiachen Li, Xin Lan, Yueqian Lin, Wengao Ye, Junwei He, Songlin Li, Yue Zhang, Yipeng Gao, Yijiang Li, Ze Ma, Liqiang Jing, Tianyu Wang, Kaixin Li, Yiqi Xue, Haoran Lyu, Yizhuo He, Yuchen Tian, Shutong Wu, Bowei Wang, Yixuan Gao, Bo Chen, Litong Liu, Sikai Cheng, Jiajun Bao, Shuaicheng Tong, Shuwen Xu, Terry Yue Zhuo, Tinghan Ye, Qi Qi, Miao Li, Longtai Liao, Zelin Tan, Chang Shi, Xilin Tang, Srinath Tankasala, Boqin Yuan, Yaoyao Qian, Jianhong Tu, Chenguang Wang, Yizhou Sun, Wei Wang, Aaron Taylor, Ziyue Yang, Changkun Guan, Zhikang Dong, Xinyu Zhang, Steven Dillmann, Han chung Lee, and Dawn Song. Skillsbench: Benchmarking how well agent skills work across diverse tasks, 2026b. URL https://arxiv.org/abs/2602.12670.

Zongxia Li, Zhongzhi Li, Yucheng Shi, Ruhan Wang, Junyao Yang, Zhichao Liu, Xiyang Wu, Anhao Li, Yue Yu, Ninghao Liu, Lichao Sun, Haotao Mi, and Leowei Liang. Long-horizon-terminal-bench: Testing the limits of agents on long-horizon terminal tasks with dense reward-based grading, 2026c. URL https://arxiv.org/abs/2607.08964.

Yusong Lin, Xinyuan Liang, Haiyang Wang, Qipeng Gu, Siqi Cheng, Jiangui Chen, Shuzhe Wu, Feiyang Pan, Lue Fan, Sanyuan Zhao, and Dandan Tu. Claw-anything: Benchmarking always-on personal assistants with broader access to user's digital world, 2026. URL https://arxiv.org/abs/2605.26086.

Jack Lindsey. Emergent introspective awareness in large language models. *arXiv preprint arXiv:2601.01828*, 2026. URL https://arxiv.org/abs/2601.01828.

Yaxi Lu, Shenzhi Yang, Cheng Qian, Guirong Chen, Qinyu Luo, Yesai Wu, Huadong Wang, Xin Cong, Zhong Zhang, Yankai Lin, Weiwen Liu, Yasheng Wang, Zhiyuan Liu, Fangming Liu, and Maosong Sun. Proactive agent: Shifting LLM agents from reactive responses to active assistance. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=sRIU6k2TcU.

Michal Lukasik, Zhao Meng, Harikrishna Narasimhan, Yin-Wen Chang, Aditya Krishna Menon, Felix Yu, and Sanjiv Kumar. Better autoregressive regression with LLMs via regression-aware fine-tuning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xGs7Ch3Vyo.

Olga Manakina, Igor Bogdanov, and Chung-Horng Lung. Delay-of-gratification as a multi-agent survival micro-benchmark for long-horizon LLMs: Social exposure, personas, and tool use budgets. In *First Workshop on Multi-Turn Interactions in Large Language Models*, 2025. URL https://openreview.net/forum?id=lGYJdWYUOf.

Davide Paglieri, Bartłomiej Cupiał, Jonathan Cook, Ulyana Piterbarg, Jens Tuyls, Edward Grefenstette, Jakob Nicolaus Foerster, Jack Parker-Holder, and Tim Rocktäschel. Learning when to plan: Efficiently allocating test-time compute for LLM agents, 2025. URL https://arxiv.org/abs/2509.03581.

Jianhao Ruan, Zhihao Xu, Yiran Peng, Fashen Ren, Zhaoyang Yu, Xinbing Liang, Jinyu Xiang, Bang Liu, Chenglin Wu, Yuyu Luo, and Jiayi Zhang. AOrchestra: Automating sub-agent creation for agentic orchestration, 2026. URL https://arxiv.org/abs/2602.03786.

Lin Shi, Haowei Lin, Zixuan Zhu, Xiaoyue Zhou, Xiang Li, Xiangning Lin, Yaxuan Deng, Han Xu, Yuangang Li, Shanda Li, Zizhao Chen, Hanwen Xing, Harsh Raj, Bo Chen, Quan Shi, Steven Dillmann, Yipeng Gao, Puneesh Khanna, Ruofan Lu, Chao Beyond Zhou, Michael Yang, Robert Zhang, Siyuan Chai, Jiayu Chang, Yizhao Chen, Xiaokun Chen, Yiwei Dai, Wenting Yang, Hange Liu, Minghao Liu, Zihan Wang, Adnan El Assadi, Benedikt Stroebl, E. Kelly Buchanan, Han Meng, Junwei He, Longxuan Yu, Radin Shayanfar, Yukyung Lee, Zhikang Dong, Allen G Hart, Anjiang Wei, Anurag Kashyap, Arpandeep Khatua, Audrey Jixin Zheng, Chengrui Ma, David Heineman, Dubing Chen, Hai-Anh Trinh, Haishuo Fang, Hefan Zhang, Hui Shen, Issa Sugiura, Jiankai Sun, Jiechao Gao, Junhong Lin, Junnan Li, Kai Yang, Lei Hsiung, Maoyu Wang, Mengze Tang, Nabil Omi, Negin Raoof, Nicholas Edwards, Octavia Guo, Orfeas Menis Mastromichalakis, Pengliang Ji, Przemysaw Hejman, Qi Qi, Qunshu Lin, Richard Zhuang, Rui Yang, Ruichen Zheng, Ryan Marten, Shaghayegh Fazliani, Shizheng Hou, Sicong Jiang, Sijie Li, Boqin Yuan, Michael Glass, Song Bian, Terry Yue Zhuo, Tianqing Wu, Tom Tang, Wanjia Zhao, Weihao Xuan, Wenhua Liang, Xian Liu, Xin Lan, Xuan Zhang, Xuandong Zhao, Yanchuan Tang, Yifan Jiang, Yijiang Li, Yitong Guan, Yizhi Li, Yonghui Liu, Yuheng Tang, Yujun, Mao, Yunfei Zhao, Yuxin Wang, Yuxuan Tang, Zhenheng Tang, Zhifei Li, Ziruo Wang, Ziyu She, Kaiyuan Liu, Iheb Chaabane, Yuxin Tang, Xiangyi Li, Satya Sai Srinath Namburi GNVV, Xinyue Zheng, Andy Konwinski, Boxuan Li, Leon Liangyu Chen, Alex Dimakis, Nicholas Carlini, Soroush Vosoughi, Sanmi Koyejo, Di He, Etash Guha, Benjamin Feuer, Mike Merrill, Ludwig Schmidt, and Alex Shaw. Harbor adapters and harbor-index: Infrastructure and a curated meta-dataset for large-scale agentic evaluation, 2026. URL https://arxiv.org/abs/2609.04298.

Tongyue Shi, Jun Ma, Zihan Yu, Haowei Xu, Rongxin Yang, Minqi Xiong, Meirong Xiao, Yilin Li, Huiying Zhao, and Guilan Kong. Large language models in critical care medicine: Scoping review. *JMIR Med Inform*, 13:e76326, Nov 2025. ISSN 2291-9694. doi: 10.2196/76326. URL https://medinform.jmir.org/2025/1/e76326.

Enxin Song, Wenhao Chai, Weili Xu, Jianwen Xie, Yuxuan Liu, and Gaoang Wang. Video-MMLU: A massive multi-discipline lecture understanding benchmark. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2025. URL https://openaccess.thecvf.com/content/ICCV2025W/Findings/papers/Song_Video-MMLU_A_Massive_Multi-Discipline_Lecture_Understanding_Benchmark_ICCVW_2025_paper.pdf.

Jakkula Sravanthi, Rajeev Sobti, Amit Semwal, M. Shravan, Aqeel A. Al-Hilali, and Malik Bader Alazzam. AI-assisted resource allocation in project management. In *2023 3rd International*

Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), pp. 70–74, 2023. doi: 10.1109/ICACITE57410.2023.10182760. URL https://doi.org/10.1109/ICACITE57410.2023.10182760.

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

Ujjwal Upadhyay, Mukul Ranjan, Zhiqiang Shen, and Mohamed Elhoseiny. Time blindness: Why video-language models can't see what humans can?, 2025. URL https://arxiv.org/abs/2505.24867.

Hjalmar Wijk, Tao Roa Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Joshua M Clymer, Jai Dhyani, Elena Ericheva, Katharyn Garcia, Brian Goodrich, Nikola Jurkovic, Megan Kinniment, Aron Lajko, Seraphina Nix, Lucas Jun Koba 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 *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=3rB0bVU6z6.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems 37*, 2024. URL https://neurips.cc/virtual/2024/poster/97468.

Weichao Xu, Huaxin Pei, Jingxuan Yang, Yuchen Shi, Yi Zhang, and Qianchuan Zhao. Exploring critical testing scenarios for decision-making policies: An LLM approach, 2024. URL https://arxiv.org/abs/2412.06684.

Haoran Zhang, Luxin Xu, Zhilin Wang, Runquan Gui, Shunkai Zhang, Haodi Lei, Zihao He, Bingsu He, Chicheng Qin, Tong Zhu, Xiaoye Qu, Yang Yang, Yu Cheng, and Yafu Li. π-bench: Evaluating proactive personal assistant agents in long-horizon workflows, 2026. URL https://arxiv.org/abs/2605.14678.

Yilun Zhao, Lujing Xie, Haowei Zhang, Guo Gan, Yitao Long, Zhiyuan Hu, Tongyan Hu, Weiyuan Chen, Chuhan Li, Junyang Song, Zhijian Xu, Chengye Wang, Weifeng Pan, Ziyao Shangguan, Xiangru Tang, Zhenwen Liang, Yixin Liu, Chen Zhao, and Arman Cohan. MMVU: Measuring expert-level multi-discipline video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/papers/Zhao_MMVU_Measuring_Expert-Level_Multi-Discipline_Video_Understanding_CVPR_2025_paper.pdf.

A Supporting Results

		Prospective			Retrospective		
Regime	**Model**	**Ratio**	**Log err.**	r	**Ratio**	**Log err.**	r
Long-horizon	GPT-5.5	2.39× [2.11, 2.94]	0.99 [0.83, 1.10]	0.26 [0.05, 0.48]	1.07× [0.99, 1.18]	0.27 [0.18, 0.38]	0.76 [0.68, 0.84]
	Opus 4.8	1.01× [0.84, 1.48]	0.81 [0.75, 0.91]	0.07 [−0.16, 0.26]	0.76× [0.59, 0.91]	0.40 [0.29, 0.54]	0.64 [0.54, 0.73]
Short agentic	GPT-5.5	7.96× [3.34, 10.15]	2.07 [1.21, 2.32]	0.70 [0.02, 0.87]	0.45× [0.27, 0.79]	0.99 [0.63, 1.30]	0.67 [0.27, 0.87]
	Opus 4.8	3.99× [3.33, 5.84]	1.38 [1.20, 1.77]	0.87 [0.24, 0.95]	1.26× [1.06, 1.71]	0.43 [0.23, 0.62]	0.92 [0.61, 0.97]
Single-turn	GPT-5.5	3.47× [2.99, 4.16]	1.24 [1.10, 1.43]	0.55 [0.33, 0.73]	0.93× [0.85, 1.05]	0.30 [0.20, 0.35]	0.67 [0.53, 0.79]
	Opus 4.8	2.87× [2.46, 3.12]	1.06 [0.90, 1.14]	0.81 [0.71, 0.87]	1.80× [1.62, 2.14]	0.59 [0.48, 0.76]	0.81 [0.69, 0.88]

Table 9: The headline results with 95% intervals from resampling tasks (5,000 draws). For reference, two runs of the same short agentic task on the same provider differ by a median factor of 1.46, so a perfect forecast of the typical run would miss a single run by about 1.30.

Prospective estimate	GPT-5.5 Ratio → 1×	GPT-5.5 Log err. ↓	GPT-5.5 Corr. r ↑	Opus 4.8 Ratio → 1×	Opus 4.8 Log err. ↓	Opus 4.8 Corr. r ↑
Zero-shot	2.05×	0.96	0.16	0.76×	0.85	0.15
One exemplar duration	1.91×	0.77	0.38	0.90×	0.67	0.31
Three exemplar durations	1.52×	0.56	0.39	1.20×	0.54	0.40
Median of the three exemplars (no model)	1.13×	0.50	0.39	0.87×	0.52	0.32
Zero-shot rescaled by one constant (no model)	1.00×	0.64	0.40	1.00×	1.06	0.14
Measured pace stated	1.18×	0.84	0.50	1.12×	0.52	0.49
Pace stated ten times too slow	4.02×	1.39	0.53	2.83×	1.07	0.19
Predicted counts through a fitted rate	0.60×	0.54	0.42	0.95×	0.72	0.18
The other model's estimate of this run	1.65×	0.64	0.35	1.11×	0.99	0.06

Table 10: Calibrating the estimate before a run on the 48 ablation tasks. Rows marked no model are simple rules that do not ask the model. Ratio is the median estimate over the actual duration, log err. the median absolute log error, and r the correlation on log durations.

This appendix collects the tables behind the results figure and the one-paragraph results in the main text, under the labels the main text cites.

Forecast of the time still ahead	Ratio	Log err. ↓	Within 2× ↑
The transcript so far	3.75×	1.34	33%
The transcript and the elapsed time	4.07×	1.43	28%
No model call: the time already spent	2.52×	1.09	33%
No model call: a typical run, less the time spent	0.99×	1.98	22%

Table 11: Forecasting the time still ahead, over 94 checkpoints inside the open model's recorded runs across four providers. Ground truth comes from the runs' own tool-call timestamps, so both the elapsed and the remaining time are measured rather than assumed.

Phase	Estimate	Ratio	Log err.	Within 2×	Tracking r
Prospective	Without the figure	11.73×	2.46	9%	−0.23
	Serving figure stated	3.68×	1.34	21%	0.35
	Rate arithmetic alone	10.28×	2.33	10%	0.50
	Rate and one fitted constant	1.01×	0.64	56%	0.50
	One fitted constant	1.02×	0.68	51%	−0.23
Retrospective	Without the figure	4.07×	1.40	23%	0.32
	Serving figure stated	1.40×	0.43	71%	0.64
	Rate arithmetic alone	3.03×	1.11	21%	0.70
	Rate and one fitted constant	1.02×	0.44	68%	0.70
	One fitted constant	1.08×	0.64	52%	0.32

Table 12: Deployment-informed estimation against calibration controls, on one open model over 19 tasks and 4 providers, 70 prospective and 75 retrospective cells, being those where both model arms answered. The contexts pass through the same execution-clock scrub as the long-horizon contexts. The indented rows use no model call: rate arithmetic scales the blind estimate by the measured decode rate, and fitted constants are medians over the other tasks, holding out all four providers of the task being scored. Tracking r correlates how far an estimate moves across providers within a task with how far the duration moves. Each cell is one run, so the durations move with an SD of 0.61 log units including repeat noise, against 0.49 with repeats averaged (Table 4).

Phase	Estimator effort	GPT-5.5 Ratio $\to 1\times$	GPT-5.5 Log err. $\downarrow$	GPT-5.5 Corr. $r \uparrow$	Opus 4.8 Ratio $\to 1\times$	Opus 4.8 Log err. $\downarrow$	Opus 4.8 Corr. $r \uparrow$
Prospective	Medium (recorded)	2.29×	0.97	0.37	0.83×	0.78	0.20
	Medium (repeat)	2.38×	1.13	0.27	0.89×	0.80	0.28
	High	2.66×	1.11	0.22	0.82×	0.83	0.24
Retrospective	Medium (recorded)	1.02×	0.37	0.61	0.61×	0.54	0.70
	Medium (repeat)	0.75×	0.30	0.72	0.72×	0.55	0.68
	High	0.67×	0.41	0.69	0.66×	0.59	0.67

Table 13: Estimator effort and repeatability on the 24-task subset (23 retrospective answers for GPT-5.5). Log err. is median absolute log error; 0 is exact and 0.69 corresponds to a factor of two. Ratios are median estimate/actual: below $1\times$ means underestimation and above $1\times$ means overestimation. r is Pearson correlation on log durations.

B LONG-HORIZON TASKS: ADDITIONAL DETAIL

B.1 ESTIMATION PROMPTS

Both questions state the same boundary on what the duration includes. The prospective question asks how long the model would take to complete the task, and Appendix B.11 shows that asking instead for the time until the attempt stops does not significantly change accuracy.

Prospective question

```
Before starting the task, estimate how long it would take YOU to
complete it using {scaffold} with task-execution reasoning effort
{effort}.
Do not execute or continue the task.
Include model inference, tool execution, and waiting during agent
execution.  Exclude environment provisioning before the agent starts,
verifier grading after it stops, and the time used to answer this
question.
Reply with ONLY: "My estimate:  X seconds" or "My estimate:  X
minutes".
```

Retrospective question

```
The recorded task attempt has ended.  Estimate how long this recorded
attempt took from the start of agent execution until it stopped,
whether or not it succeeded.
Do not execute or continue the task.
Include model inference, tool execution, and waiting during agent
execution.  Exclude environment provisioning before the agent starts,
verifier grading after it stops, and the time used to answer this
question.
Reply with ONLY: "Time taken:  X seconds" or "Time taken:  X
minutes".
```

B.2 RETROSPECTIVE CONTEXT

The retrospective question gives the context the agent held at the point it stopped, including any summary or memory blocks the scaffold had already written into that view. Where a scaffold shortened its history more than once, we apply the recorded edits in order and use the last request the benchmark recorded. We write no summaries of our own. A run whose stopping context cannot be reconstructed is reported as missing.

B.3 TASKS BY SOURCE AND CATEGORY

SkillsBench (34). *Software and data-system engineering:* `debug-trl-grpo`, `fix-build-google-auto`, `fix-druid-loophole-cve`, `fix-visual-stability`, `flink-query`, `parallel-tfidf-search`, `setup-fuzzing-py`. *Scientific analysis and ML:* `earthquake-phase-association`, `exoplanet-detection-period`, `flood-risk-analysis`, `glm-lake-mendota`, `lab-unit-harmonization`, `quantum-numerical-simulation`, `seismic-phase-picking`, `shock-analysis-demand`, `shock-analysis-supply`. *Planning, optimization and control:* `adaptive-cruise-control`, `bike-rebalance`, `civ6-adjacency-optimizer`, `drone-planning-control`, `energy-unit-commitment`, `exam-block-sequencing`, `grid-dispatch-operator`, `manufacturing-fjsp-optimization`. *Professional analytical work:* `energy-market-pricing`, `manufacturing-equipment-maintenance`, `reserves-at-risk-calc`. *Documents, media and CAD artifacts:* `ada-bathroom-plan-repair`, `court-form-filling`, `jpg-ocr-stat`, `latex-formula-extraction`, `organize-messy-files`, `paper-anonymizer`, `video-silence-remover`.

Harbor-Index (11). *Software and data-system engineering:* `algotune-optimize-outer-product`, `swebenchpro-fix-file-suffix-chooser`, `swebenchpro-`

fix-teleport-k8s-session-path, swebenchpro-fix-teleport-mtls-ca-limit, swebenchpro-improve-process-signal-messages, swebenchverified-fix-django-union-queryset-order, tb-make-doom-for-mips. *Scientific analysis and ML:* dacode-predict-essay-scores, hle-shock-wave-density-profile, replicationbench-find-galactic-vz-peaks, sldbench-discover-vocab-scaling-law.

DeepSWE (38). *Software and data-system engineering:* abs-stepped-slices, actionlint-action-pinning-lint, anko-typed-variable-bindings, awilix-async-container-initialization, bandit-interprocedural-taint-checks, cattrs-partial-structuring-recovery, clack-async-autocomplete-options, cliffy-config-file-parsing, csstree-shorthand-expansion-compression, dynamodb-toolbox-lazy-recursive-schemas, goreleaser-retry-publish-auditing, gql-incremental-graphql-delivery, happy-dom-deterministic-intersectionobserver, helm-array-merge-strategies, httpx-streaming-json-iteration, ipython-session-bundle-replay, kea-atomic-signal-selectors, kombu-single-active-consumer-priority, kombu-virtual-queue-dead-lettering, koota-composite-trait-aspects, koota-entity-snapshot-rollback, koota-pair-relation-tracking, kysely-window-grouping-helpers, numba-stencil-boundary-modes, obsidian-linter-auto-table-of-contents, opa-template-string-reconstruction, oxvg-structural-selector-preservation, psd-tools-blend-range-api, quill-shared-toolbar-focus, scc-bounded-memory-spilling, sql-formatter-bigquery-pipe-formatting, tengo-callable-instance-isolation, true-myth-iterable-collection-combinators, ts-pattern-match-each, updo-policy-alerting, valibot-recursive-schema-composition, wasmi-trap-coredumps, ytt-jsonpath-query-api.

LHTB (6). *Professional analytical work:* apex-ib244-matter, apex-law433-matter, apex-management-consulting-matter. *Software and data-system engineering:* commit0-multilib-tdd. *Planning, optimization and control:* rush_hour_campaign. *Scientific analysis and ML:* tabular-data-feature-covshift.

Toolathlon-Verified (6). *Documents, media and CAD artifacts:* interview-report, vlm-history-completer. *Professional analytical work:* live-transactions, nvidia-market. *Business and e-commerce operations:* canvas-art-manager, woocommerce-new-product.

B.4 Benchmarks considered

We also screened ProgramBench, OSWorld 2.0, WeaveBench, WildClawBench, RUBench, ResolveBench and ResearchArena. Each fell short on at least one of the requirements above: released runs for only one of the two models, no per-run wall-clock timing, a time cap that ends most runs, or gated access to the trajectories. They remain candidates for a later extension, and OSWorld 2.0 in particular would add desktop GUI work, which the current set lacks.

B.5 Long-horizon results by source

Table 14 breaks the long-horizon results of Table 2 down by source benchmark. Two GPT-5.5 retrospective answers on LHTB are absent because their stopping contexts exceed the deployment's request limit, so the pooled GPT-5.5 row uses 93 tasks.

B.6 Short agentic and single-turn results in detail

Table 15 gives the short agentic results for each task family and the single-turn results with mean absolute error in seconds.

				Prospective			Retrospective		
Source	**Model**	n	**Actual** min	**MAE (s)** ↓	**Ratio** → 1×	**Corr.** r ↑	**MAE (s)** ↓	**Ratio** → 1×	**Corr.** r ↑
SkillsBench	GPT-5.5	34	8.3	1664	3.08×	0.11	161	1.12×	0.78
	Opus 4.8	34	24.1	912	0.62×	0.11	730	0.44×	0.83
Harbor-Index	GPT-5.5	11	18.9	806	0.91×	0.20	603	0.46×	0.91
	Opus 4.8	11	58.4	2250	0.33×	0.71	2178	0.33×	0.80
DeepSWE	GPT-5.5	38	19.0	1888	2.45×	0.49	611	1.42×	0.77
	Opus 4.8	38	11.2	992	2.19×	0.20	275	1.14×	0.72
LHTB	GPT-5.5	4	10.6	5771	8.44×	0.97	204	0.89×	0.96
	Opus 4.8	6	39.9	4165	2.52×	0.87	810	0.80×	0.97
Toolathlon	GPT-5.5	6	10.2	459	0.56×	0.59	185	0.77×	0.09
	Opus 4.8	6	23.5	1042	0.17×	−0.66	652	0.47×	0.82
All	GPT-5.5	93	13.4	1753	2.39×	0.26	401	1.07×	0.76
	Opus 4.8	95	18.1	1312	1.01×	0.07	716	0.76×	0.64

Table 14: Long-horizon estimation by source. Actual is median recorded duration; MAE is mean absolute error in seconds (lower is better). Ratios are median estimate/actual: below 1× means underestimation and above 1× means overestimation. r is Pearson correlation on log durations.

				Prospective			Retrospective		
Set	**Model**	n	**Actual** s	**MAE (s)** ↓	**Ratio** → 1×	**Corr.** r ↑	**MAE (s)** ↓	**Ratio** → 1×	**Corr.** r ↑
Custom agentic	GPT-5.5	11	47.5	269	6.72×	−0.46	41.4	0.34×	0.45
	Opus 4.8	11	33.7	123	4.62×	−0.04	13.6	1.33×	–
METR public	GPT-5.5	8	75.0	1026	8.33×	0.87	61.9	0.63×	0.70
	Opus 4.8	8	61.7	451	3.35×	0.88	91.9	1.26×	0.91
All short agentic	GPT-5.5	19	57.6	588	7.96×	0.70	50.1	0.45×	0.67
	Opus 4.8	19	39.0	261	3.99×	0.87	46.6	1.26×	0.92
Single-turn	GPT-5.5	68	12.1	92.2	3.47×	0.55	4.4	0.93×	0.67
	Opus 4.8	68	15.2	30.5	2.87×	0.81	16.1	1.80×	0.81

Table 15: Short agentic and single-turn estimation. Actual is the median recorded duration in seconds, MAE is in seconds. Opus 4.8 returned the same retrospective answer, 45 seconds, on every custom agentic task, so its correlation there is undefined. Both models solve nearly every short agentic task (Opus 4.8 18 of 19, GPT-5.5 19 of 19). Ratios are median estimate/actual: below 1× means underestimation and above 1× means overestimation. r is Pearson correlation on log durations.

B.7 Context fidelity and recorded outcome

The long-horizon sources release their runs at different levels of detail. SkillsBench releases the provider's own request records, which we use as recorded for the Opus 4.8 runs with the hidden reasoning state removed; the GPT-5.5 runs and every other source use the released transcript. Thirteen SkillsBench tasks also have a GPT-5.5 answer from the recorded request, which measures how much the context condition matters (Table 16). Splitting the main panel by the released outcome of each run (Table 17) shows solved and failed runs estimated alike; no admitted run stopped at a time cap.

B.8 Workload prediction by quantity

Table 18 compares the work each model predicts before a run with the work the run recorded, for each counted quantity.

Phase	Condition	Ratio $\to 1\times$	Log err. $\downarrow$	Corr. $r \uparrow$	Identical
Prospective	Recorded request	2.29×	0.83	0.18	0/13
	Released transcript	3.04×	1.11	−0.18	
Retrospective	Recorded request	0.93×	0.40	0.62	4/13
	Released transcript	1.11×	0.13	0.81	

Table 16: GPT-5.5 on the 13 SkillsBench tasks with answers under both context conditions. Transcript and recorded-request answers agree within a median 18% retrospectively and 34% prospectively. Ratios are median estimate/actual: below $1\times$ means underestimation and above $1\times$ means overestimation. Log err. is median absolute log error; 0 is exact and 0.69 corresponds to a factor of two. r is Pearson correlation on log durations. Identical counts matching answers across the two context conditions.

			Prospective			Retrospective		
Model	Outcome	n	Ratio $\to 1\times$	Log err. $\downarrow$	Corr. $r \uparrow$	Ratio $\to 1\times$	Log err. $\downarrow$	Corr. $r \uparrow$
GPT-5.5	Solved	45	2.25×	0.96	0.28	1.07×	0.23	0.81
	Failed	44	2.38×	0.98	0.23	1.16×	0.34	0.71
Opus 4.8	Solved	43	0.84×	0.81	0.06	0.67×	0.40	0.55
	Failed	46	1.00×	0.80	−0.27	0.86×	0.37	0.64

Table 17: Long-horizon estimation by released outcome. Six LHTB runs release a stop status without a pass label and are omitted. Ratios are median estimate/actual: below $1\times$ means underestimation and above $1\times$ means overestimation. Log err. is median absolute log error; 0 is exact and 0.69 corresponds to a factor of two. r is Pearson correlation on log durations.

Model	Quantity	Labelled	Ratio $\to 1\times$	Log err. $\downarrow$	Corr. $r \uparrow$
GPT-5.5	Model requests	32	0.55×	0.60	0.61
	Tool calls	47	0.55×	0.69	0.50
	Completion tokens	22	2.97×	1.10	0.38
Opus 4.8	Model requests	48	0.97×	0.28	0.76
	Tool calls	48	0.87×	0.31	0.61
	Completion tokens	22	2.69×	1.19	−0.21

Table 18: Prospective prediction of the run's own workload on the 48 ablation tasks. Labelled gives the number of tasks whose released records carry that quantity; request counts are absent for GPT-5.5 on SkillsBench and completion tokens exist for DeepSWE, LHTB and Toolathlon only. Ratios are median estimate/actual: below $1\times$ means underestimation and above $1\times$ means overestimation. Log err. is median absolute log error; 0 is exact and 0.69 corresponds to a factor of two. r measures correlation between predicted and recorded workload.

B.9 Serving-speed measurements

We measure each provider's serving speed separately from the agent runs, with fixed prompts, so that no run's own timing enters the speed it is compared against (Table 19).

Provider	Decode (tok/s) ↑	TTFT (s) ↓	Blocks 1 / 2 / 3 (tok/s)	$ per M in / out
Parasail	209	0.8	262 / 186 / 192	1.40 / 4.40
Baidu	94	0.9	88 / 96 / 95	0.77 / 2.43
Z.AI	78	2.9	83 / 78 / 78	1.40 / 4.40
Novita	64	1.7	68 / 67 / 58	0.87 / 2.72
SiliconFlow	79	1.7	76 / 79 / 87	1.12 / 3.52
Baseten	145	1.3	70 / 142 / 289	2.10 / 6.60

Table 19: GLM-5.3 at fp8 through pinned providers, measured with fixed prompts of four shapes streamed in three time blocks. Decode rate is completion tokens over the span from first to last streamed token. The deployment study uses four of these providers, spanning 64 to 209 tokens per second; Baseten's rate varied fourfold across blocks. TTFT is time to first token; arrows indicate higher throughput and lower latency.

B.10 Deployment pilot by provider

Table 20 breaks the deployment study down by provider.

		Measured serving				Median ratio	
Provider	Runs	tok/s	TTFT (s)	Median s	Solved	Pre	Post
Parasail	53	209	0.8	12.2	58%	10.39×	8.10×
Baidu	55	94	0.9	19.9	60%	6.06×	4.08×
Z.AI	54	78	2.9	43.9	63%	2.91×	2.03×
Novita	57	64	1.7	29.7	60%	4.54×	2.93×

Table 20: The deployment pilot by provider, 219 completed runs of the 228 planned. Median s is the median execution time of a run and the ratios are medians of estimate over actual. Solve rates stay within five points of each other, so the providers are running comparable work at different speeds. Because the estimates hardly move while the durations do, the same model drifts from overestimating by ten times on the fastest provider to three times on the provider with the longest runs, which also has the slowest first token.

B.11 What the prospective question asks for

The prospective question asks how long it would take the model to complete the task, while the scored duration is how long the attempt ran, including attempts that stopped without completing. To check that the two are answered alike, we ask the same tasks again with the boundary stated explicitly, in the wording the retrospective question already uses: how long the attempt runs until it stops, whether or not it succeeds. Nothing else changes.

Stating the boundary moves the estimates up, by a median factor of 1.50 for GPT-5.5 and 1.20 for Opus 4.8, and does not significantly change accuracy on these 48 tasks. GPT-5.5 goes from a median absolute log error of 0.93 to 1.23 and Opus 4.8 from 0.85 to 0.79, with paired intervals of $[-0.07, +0.46]$ and $[-0.37, +0.05]$ that both contain zero. Together with the finding that solved and failed runs are estimated alike (Table 17), this indicates that the wording of the boundary is not what limits prospective accuracy.

B.12 Robustness of the ordering result

Pooling five benchmarks of unequal size can carry a correlation on differences between benchmarks rather than within them, and SkillsBench and DeepSWE together supply 72 of the 95 tasks. Table 21 recomputes the pooled correlations three further ways: dropping each source in turn, averaging the per-source correlations so that benchmarks rather than tasks carry equal weight, and removing each source's mean log duration before pooling. The retrospective result is stable throughout. The prospective result is weak in every version, and for Opus 4.8 the pooled 0.07 understates what remains inside a benchmark, which is 0.25 to 0.30.

Model	Phase	Pooled	Source-centred	Mean within	Leave one source out	
					Lowest	Highest
GPT-5.5	Prospective	0.26	0.32	0.47	0.19	0.47
	Retrospective	0.76	0.79	0.70	0.65	0.84
Opus 4.8	Prospective	0.07	0.30	0.25	−0.08	0.35
	Retrospective	0.64	0.79	0.83	0.59	0.81

Table 21: The long-horizon correlation on log durations, recomputed so that differences between source benchmarks cannot carry it. Source-centred removes each benchmark's mean before pooling and mean within averages the per-benchmark correlations. The last two columns give the range over the five leave-one-source-out fits. Prospective figures use the tasks that also carry a retrospective answer, 93 for GPT-5.5 and 95 for Opus 4.8, so that the two phases are compared on the same runs; over all 95 tasks GPT-5.5's prospective correlation is 0.33.

C Calibration by Training on an Open Model

Training a model to estimate its own run time requires access to its weights, so we study calibration by training on Qwen3-8B, an open model that we run locally on a single A100 GPU. Running locally also lets us time every generation directly and repeat executions cheaply, which training needs. We use the 68 single-turn tasks of Section 3 as the held-out benchmark and a separate pool of 132 single-turn tasks for training, so that no evaluated task is seen during training.

Setup. For each task, the model is asked before generating how long it will take to complete the task, and after generating how long its response took, and we time the generation from its start to its last token. Qwen3-8B runs with `max_tokens=2048`, a repetition penalty of 1.1 and the temperature its model card recommends. Every method below is scored on the same 68 held-out tasks with mean absolute error in seconds and Pearson correlation.

Estimate before generating

```
Task: {description}
How long will it take you to complete this? Give a specific estimate
in seconds.
Reply with just the number and unit, e.g. "30 seconds" or "2
minutes".
```

Estimate after generating

```
You just completed this task: {description}
Your response: {output}...
How long did it take you to generate that response?
Give your best estimate in seconds. Reply with just the number and
unit.
```

Methods. We compare five ways of training the estimate, all starting from the same base model. *Supervised fine-tuning* trains on the measured durations of the training tasks, either from the plain prompt or with the model's measured generation speed of 16.63 tokens per second added to the prompt. A *regression head* is a linear layer on the last hidden state that predicts the duration directly, without generating text. *RAFT* (Lukasik et al., 2025) scores a grid of candidate durations with the model's own likelihood and predicts their probability-weighted average,

$$\hat{t}_{\text{RAFT}}(x) = \sum_{t_k \in \mathcal{T}} p(\text{str}(t_k) \mid x)\, t_k,$$

over a log-uniform grid $\mathcal{T}$ from 1 to 600 seconds, and fine-tunes the model to bring this average close to the measured duration; the asymmetric variant penalizes overestimates more than underestimates. *Reinforcement learning* rewards the model for estimates close to the measured duration, either for a structured answer or after a short reasoning trace.

		Before generating		After generating	
Type	**Method**	**MAE (s)** $\downarrow$	$r \uparrow$	**MAE (s)** $\downarrow$	$r \uparrow$
No training		125.77	.31	54.32	.27
SFT	Speed stated	30.45	.39	3.22	.99
	Plain prompt	34.32	.30	27.16	.63
Regression	Linear head	24.70	.21	21.34	.10
RAFT	Asymmetric	24.04	.44	25.32	.54
	Symmetric	26.30	.41	–	–
RL	Structured	61.84	.09	33.25	.47
	With reasoning	43.77	.28	23.31	.37

Table 22: Calibration by training on Qwen3-8B, scored on the 68 held-out single-turn tasks. MAE is mean absolute error in seconds and r the Pearson correlation between estimated and measured duration.

Results. Every training method reduces the error before generating, from 126 seconds without training to between 24 and 62 seconds (Table 22). How well the estimates rank tasks improves far less, since the best correlation before generating is 0.44, for RAFT. The regression head shows why the two metrics must be read together: it reaches a low error by predicting nearly the same duration for every task, so its correlation stays near zero. After generating, fine-tuning with the stated speed is almost exact, with an error of 3 seconds and a correlation of 0.99. In this setting the prompt contains both the response and the speed, so the model has learned the arithmetic of dividing the length of its response by its speed. Training therefore teaches a model the scale of its own run times and how to use a stated speed, while forecasting which tasks will take longer remains the harder problem, in line with the main results.

Training details. Fine-tuning uses LoRA with rank 16 and alpha 32, a learning rate of 2×10^{-4}, batch size 4 and three epochs. The regression head is a single linear layer on the last hidden state, trained with squared error on log durations. Reinforcement learning uses a reward based on log-scale estimation error with a KL penalty of 0.05. RAFT needs one forward pass per candidate duration, which makes each estimate slower than free generation.