LEAPS: LONG-CONTEXT EVALUATION OF ADAPTIVE PATTERN LEARNING AND STATE TRACKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context language models (LCLMs) power agents that act over long trajectories, from improving a codebase to running research experiments to forecasters that condition on lengthy histories. These applications draw on three distinct long-context capabilities: recall, where the answer is a fact present in the context; state tracking, where the outcome can be computed by performing a series of transformations to a specified state; and pattern learning, where the goal is to make a prediction by sampling from a distribution inferred from many observations. Existing benchmarks primarily focus on the first capability. In this work, we introduce **LEAPS** (Long-context Evaluation of Adaptive Pattern learning and State tracking), a collection of 12 novel datasets, spanning agentic workflows, behavioral modeling, and forecasting of market and world events. By construction, LEAPS is resistant to data contamination, extends to millions of input tokens, and uses reliable metrics. We evaluate 30 API and open models from 8K to 512K input tokens. LEAPS distinguishes models that perform near perfectly on recall tasks, finding that such models' state tracking and pattern learning abilities can vary significantly and degrade much more dramatically with increasing input lengths—even for pattern learning, where more observations should make learning a theoretically easier task. Moreover, thinking models often underperform their base counterparts, exhausting their generation budget or oversimplifying the task. We hope LEAPS can guide the development of the next generation of LCLMs.[1]

1 INTRODUCTION

Today, long-context language models (LCLMs) power personal assistants, coding agents, problem solving algorithms, and a wide variety of other tasks. Such applications frequently require comprehending large amounts of context (Kwa et al., 2025; Yang et al., 2024; Wang et al., 2025) specific to the task and rely on the underlying model having multiple distinct *long-context capabilities*.

For example, in order for a coding agent to successfully autonomously edit files and run commands over a long trajectory, it must be able to perform several sub-tasks. It should be able to locate code relevant to the current step, which relies on a capability commonly called *recall* (Arora et al., 2024; Hsieh et al., 2024). It must also be aware of the current state of the repository and its progress on each objective after many edits, requiring strong *state tracking* and understanding (Liu et al., 2023; Merrill & Sabharwal, 2023). Finally, its actions in unseen situations must align with the user's preferences, which it must infer from their distribution of past interactions. We call this type of in-context learning ability *pattern learning* (Brown et al., 2020; Garg et al., 2022). Many long-horizon tasks, particularly harder ones, frequently make use of all three abilities, which fundamentally differ in *how* models use their available context: to recall facts, maintain and update a relatively comprehensive (but compact) state as changes are applied, or learn patterns from observations.

However, existing LCLM development still overwhelmingly relies on recall benchmarks, such as Needle-In-A-Haystack (Kamradt, 2023) and RULER (Hsieh et al., 2024). Some recent benchmarks that evaluate performance on a broader range of realistic applications, such as question-answering, summarization, and reasoning over code (Zhang et al., 2024; Yen et al., 2025; Bai et al., 2025, *inter alia*), many-shot in-context learning (Agarwal et al., 2024; Bertsch et al., 2025a; Li et al., 2025),

[1] Code and data will be available upon publication.

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.

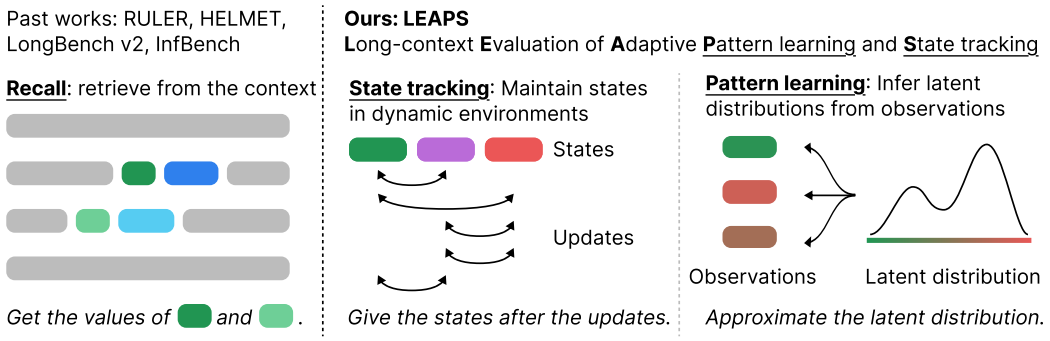

Figure 1: In contrast to many previous long-context benchmarks that revolve around *recall*, we study two new capabilities in a long-context setting, *state tracking* and *pattern learning*.

and aggregation (Bertsch et al., 2025b) do broaden the coverage of long context evaluation, but do not comprehensively study how a model's relative long context capabilities are apportioned between recall, state tracking, and pattern learning.

To fill this gap, we systematically study two long-context capabilities—state tracking and pattern learning—by creating a novel benchmark, **LEAPS** (Long-context Evaluation of Adaptive Pattern learning and State tracking).We construct 6 synthetic and 6 real-derived datasets in which LCLMs track ML experiments, imitate top chess players' styles, and forecast events from historical data. By design, our datasets are robust to contamination (these datasets can be regenerated or sampled past a model's knowledge cutoff), scale to millions of input tokens, and use reliable evaluation metrics.

Our experiments cover 30 API and open models, evaluated from 8K to 512K input tokens. Even models that look very similar on recall tasks can be starkly distinguished on LEAPS. We attribute this in part to LEAPS tasks testing for distinct long-context capabilities and to performance with respect to such capabilities still being relatively unsaturated, even amongst very strong API models.

In contrast to many recall benchmarks, where model performance is relatively consistent across context lengths, performance on state-tracking tasks degrades significantly at long context. Performance on pattern-learning tasks—which become higher signal-to-noise with context length—also declines at long contexts (although sometimes performance can improve at moderate context).

Studying the correlation of performance in the LEAPS suite across models and context lengths suggests that our datasets evaluate multiple distinct long-context capabilities, not just performance along a single hidden axis. Further, when comparing models, a somewhat surprising observation is that thinking models often perform worse on LEAPS tasks than their base counterparts. Our analysis suggests that they often run out of generation budget without making appreciable progress and can make overly simplifying assumptions, suggesting possible improvements in long-context post-training techniques. We hope LEAPS can help develop LCLMs for further long-horizon applications.

2 OUR BENCHMARK: LEAPS

Many applications require the model to reason over complex contextual information. In agentic settings, models need to keep track of the goals and the state of the environment over long trajectories—a form of *state tracking*. In predictive settings, models need to leverage historical observations to approximate latent distributions or user behaviors—an instance of *pattern learning*. Many long-horizon tasks require a combination of these long-context capabilities, such as a coding agent that has to interact with a file system and the shell while aligning its actions with the user's preferences.

Previous benchmarks often revolve around recall (Kamradt, 2023; Hsieh et al., 2024; Bai et al., 2025; Yen et al., 2025), but state tracking and pattern learning are conceptually distinct because the answer is not present in the context. Instead, models must compute the answer by using the entire context, either composing state-modifying transformations or inferring patterns from historical data (Figure 1). Interestingly, as input length increases, state-tracking tasks remain perfectly solvable and require a constant sized memory that does not depend on the trajectory length if done efficiently, whereas the ceiling of performance on pattern learning tasks increases as more data is available.

State tracking	Description	Source
Cup swapping	Track positions of balls as cups are swapped.	Syn
Coins	Track the count of piles as coins are added, removed, and swapped.	Syn
Orderbook	Track buy and sell orders in a multi-actor market.	Syn
ML Dev	Follow parallel model developments and generate subsequent commands.	Syn
Commit re-ordering	Identify pairs of conflicting commits.	Real
Code tracking	Track code files across commits and changes.	Real
Pattern learning	Description	Source
Rock paper scissors	Learn behaviors of two players following specific strategies.	Syn
Review ranking	Learn preferences of an area chair ranking papers based on reviews.	Syn
Kaggle	Learn relationships between abstract features and market movements.	Real
Social media	Learn relationships between textual features and market movements.	Real
Chess	Learn behaviors of top chess players and predict moves at new states.	Real
Forecast	Forecast real-world events using news articles and past events.	Real

Table 1: Description and data source of each dataset (synthetic or real).

We introduce LEAPS, a suite of benchmarks which consists of six state-tracking and six pattern-learning tasks, including a mix of synthetic tasks, designed to isolate a narrow capability, and application-centric tasks highlighting a model's performance in downstream settings. We design the tasks with the following desiderata: (1) robustness to data contamination by generating data after the model knowledge cutoff, (2) scalability to millions of tokens for comparisons across lengths, (3) reliable and interpretable metrics, standardized between 0 and 1 with higher being better, (4) naive baselines for contextualizing model performances,[2] and (5) cheap inference costs. An overview of the datasets is in Table 1 and more details for all datasets are in §A.1.

2.1 STATE TRACKING

Interactive environments can be formulated as semiautomata, consisting of a set of finite states Q, such as the positions of balls or the content of a file, that are modified by a set of transformations M, such as swapping two cups or editing a line of code. We define state tracking as the ability to compute the state q_T after transformations $\mathcal{M} = \{m_1, \ldots, m_T\}$ on the initial state q_0. Recent works have shown limitations of existing models on state tracking (Merrill & Sabharwal, 2023; Merrill et al., 2024; Liu et al., 2023), and we extend the evaluation to diverse long-context settings.

Individual steps are simple and the number of bits needed to represent q_i remains bounded.[3] The task becomes increasingly more difficult as more transformations are composed together, because missing just one update can derail the entire computation, but a human can still perfectly solve all tasks with enough time. Thus, we test how efficiently models can represent and answer questions about q_T as T increases. We briefly describe each task below and details are in §A.1.

Cup swapping. In this classic game, balls are placed under exactly one cup, and the task is to maintain the position of the balls as the cups are swapped around. The input length naturally scales with more operations, and synthetic generation means the model cannot rely on its parametric knowledge for getting the correct final positions. The metric is the F1 score.

Coins. In this task, we test both swapping and linear aggregation operations. Across many piles of coins, the model must keep track of the count of coins in each pile as coins are added, removed, and swapped between piles. Similarly, the input length scales with more operations, and the data is synthetically generated. The metric is one minus the normalized L1 distance.

Orderbook. We test if models can track and reason about states in a market. We simulate multiple actors sending buy, sell, and cancel orders to a central limit order book, and trades occur as orders arrive.[4] The task is to accurately calculate how many shares of a new order would trade and at what

[2]We notice that weaker models may underperform basic strategies, such as guessing a random option, so we provide these baselines as an anchor point.

[3]Code repositories can be arbitrarily long, but we limit the number of files and lines in each task.

[4]The orderbook can grow arbitrarily large, but most orders get canceled or filled by construction.

average price. The horizon scales with the number of orders and the data is synthetically generated. We report one minus the L1 distance.

Code Tracking. Coding agents must keep track of how the code in a repository changes over time to effectively accomplish user tasks. Thus, we simulate agentic trajectories with commits from GitHub repositories.[5] During evaluation, we show the model the initial state of files and subsequent commits and ask the model to output 3–5 lines in the final state of the code. The metric is exact match, and T scales with the number of commits.

Commit Re-ordering. Coding agents are also commonly used to detect bugs and conflicts. Using the real-world commits from Code Tracking, we construct sequences of commits in which one pair of consecutive commits is out of order. The two commits modify overlapping lines, resulting in conflicts if applied out of order, and the model has to identify this pair. The metric is exact match.

ML Model Development. Long-horizon tasks involve progressing on parallel goals while following user instructions. We simulate this in an ML model development process with a synthetic shell environment. The prompt includes the specifications and goals, environment instructions, and a history of executed commands. The task is producing the next command that advances a specific goal, and the metric is exact match with any valid and useful command (definitions are in §A.1).

2.2 PATTERN LEARNING

Real-world events often generate concrete observations from latent distributions $y \sim p_\theta(\cdot \mid x)$, such as a player's chess move at a board state or market movements in response to a piece of news. We define pattern learning as the ability to approximate p_θ from observations $\{(x_i, y_i)\}_{i=1}^n$ by predicting y' for a new x'. Pattern learning powers many applications, such as forecasting future events based on recent data (Halawi et al., 2024; Karger et al., 2025) and making decisions based on user preferences in unseen situations (Park et al., 2023; Wang et al., 2023) by using numerous observations, making it a key long-context capability to study.

In-context learning tasks are similar in that they also learn from demonstrations (x, y) (Brown et al., 2020; Xie et al., 2022); however, they are limited to classification tasks and learning the output format (Min et al., 2022; Bertsch et al., 2025a; Li et al., 2025; Yen et al., 2025). LEAPS is unique in that the observations (x, y) and latent distributions p_θ are *specific* to each test instance, whereas previous benchmarks draw (x, y) from a pool of shared demonstrations. We consider diverse latent distributions, spanning player strategies to real market movements. Notably, pattern learning is distinct from recall and state tracking in that holding context degradation aside, performance of strong LCLMs should *improve* with longer inputs, as there are more data to learn from.

Rock-paper-scissors (RPS). Humans often exhibit predictable strategies in RPS (Dyson et al., 2016; Zhang et al., 2021; Wang et al., 2014). We generate synthetic players with random parameterizations that dictate the distribution over their moves, p_θ. The task is to approximate p_θ given a history of games, which scales with the input length. The metric is 1 minus the Hellinger distance.

Chess. We study chess as another environment where players have implicit preferences due to their style of play, and we use real games from top players on online websites.[6] The evaluation input consists of a long history of complete games from a player p and one incomplete game at a board state that p has never encountered before. We ask the LLM to output what move p will play next, and report the exact match score as the metric.

Review ranking. Human preferences result in comparisons and rankings of different items, and the ability to collate preference information into relative rankings is a critical skill in information retrieval (Bajaj et al., 2016; Nogueira & Cho, 2020). To this end, we design a review ranking dataset by generating "area chairs" (AC) with random preferences over five axes: Novelty, Significance, Soundness, Presentation, and Reproducibility. The input includes examples of AC's ranking over

[5] We use repositories with active changes in 2026 and permissive licensing, listed in Table 3.

[6] The games were played from January through May 2026 on `www.lichess.org`, a platform popular among top chess players.

papers, each with multiple synthetically generated reviews, and a set of new papers that the model has to rank. The metric is NDCG@20.

Kaggle. Forecasting requires inferring relationships between variables, such as how overall market movements affect a specific stock. We use a popular time-series market forecasting Kaggle competition (Desai et al., 2024) that requires using abstract features to predict ticker movements. We formulate this as a multiple-choice QA task with five approximately equally weighted bins for ease of evaluation. The metric is exact match (EM).

Social media market forecasting. Social media posts reflect public sentiment toward markets and can indicate future movements. We use an internal collection of social media posts, each matched to a specific tradable asset, to evaluate whether models can learn relationships between textual features and market movements. The prompt consists of posts and subsequent market movements, and the task is to forecast movements on a new set of posts. Similarly to Kaggle, we construct the task as a multiple-choice QA task and use the same EM metric.

General event forecasting. In addition to the stock market, we also consider a wider range of forecasting tasks, such as world events and macroeconomic trends. We use the May 2026 questions from Karger et al. (2025) and pair each question with relevant news articles from Common Crawl. The prompt includes examples from an earlier date, each with news articles, a query, and a resolution, followed by the new question and articles. The model is then asked to output the predicted probability that the query will resolve to true. We report one minus the Brier score.

3 RESULTS

Our evaluations span 30 models from 8K to 512K input tokens. Across context lengths, LEAPS distinguishes models that appear similar on recall benchmarks by testing for distinct long-context capabilities, and performance on them is still unsaturated. Even the strongest models fall short of the ceiling on state tracking and open models often cannot learn from observations effectively in pattern learning. As the context length increases, existing models experience statistically significant performance degradation on state tracking and pattern learning. Finally, post-trained models often underperform the base models they are trained from, suggesting that current post-training techniques, which target improving reasoning abilities, can compromise other long-context capabilities.

3.1 EXPERIMENT SETTINGS

We evaluate frontier API models, GPT 5.6 Sol and Terra (OpenAI, 2026), Claude Opus 4.8 and Sonnet 5 (Anthropic, 2026a;b), and Gemini 3.5 Flash (Google DeepMind, 2026), and open-weight model families, Llama 3.1 (Grattafiori et al., 2024), ProLong (Gao et al., 2025), GPT-OSS (OpenAI et al., 2025), Qwen 3 and 3.5 (Yang et al., 2025; Team, 2026), Gemma 4 (Team et al., 2026), Nemotron Nano (NVIDIA et al., 2025), and Kimi Linear (Team et al., 2025).

Our evaluation covers input lengths from 8K up to 512K tokens, measured in the Llama 3 tokenizer (Grattafiori et al., 2024).[7] For base models, we ensure the generation budget is sufficiently large to output the correct answer in each dataset. For instruction-tuned models, we increase the generation budget by 256 tokens to account for assistant messages. For thinking models, we increase the budget by 16K tokens to allow long thinking traces, and instruct the model to finish the chain-of-thought within the allotted budget in the system prompt. We instruct models not to rely on code to solve state tracking, because we want to test if models can do this task with their native representations, which is required in many less-structured real-world applications. Details are in §A.2.

3.2 LEAPS EVALUATES DISTINCT LONG-CONTEXT CAPABILITIES

LEAPS distinguishes frontier models. We present the results of frontier API models in Figure 2. The input lengths are 512K tokens, and we use medium reasoning effort for all models, when appli-

[7] Inputs with 512K tokens often translate to more than 600K tokens for many API models. We also release data at 1M tokens but did not evaluate them to leave sufficient context window for generating reasoning traces.

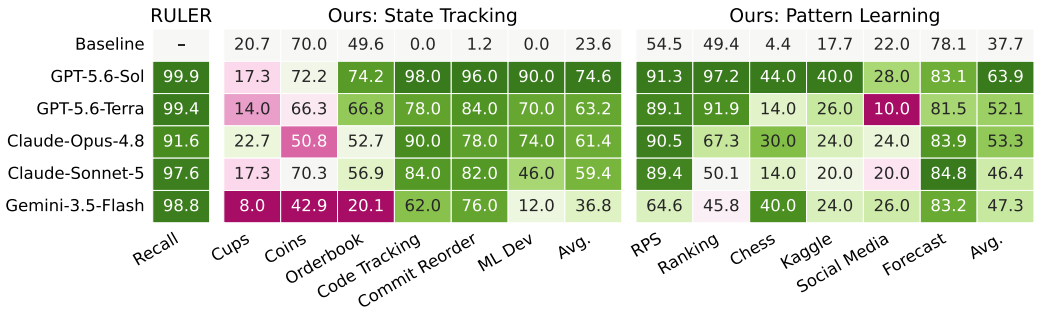

Figure 2: Results of API models at input length 512K in percentages. Compared with recall tasks, consisting of subsets of RULER and HELMET, LEAPS distinguishes models on two new axes.

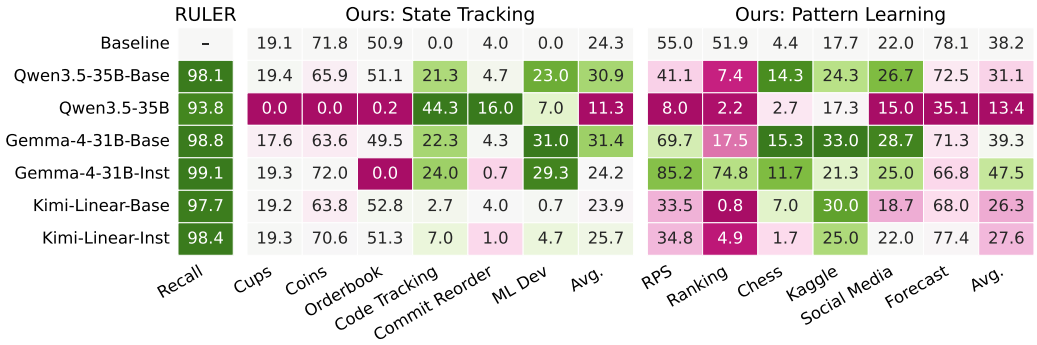

Figure 3: Results of selected open-weight models at 128K input length. Even the strongest open models struggle with both state tracking and pattern learning.

cable. We evaluate on 50 samples from each dataset.[8] Although API models have roughly saturated recall tasks, their performance varies substantially on LEAPS, allowing for a more nuanced comparison of their abilities on these two long-context capabilities. Our findings are complementary to results on other benchmarks, including GraphWalks, MRCR, and LongBench v2 (OpenAI, 2025; Vodrahalli et al., 2024; Bai et al., 2025) (§A.5). Somewhat surprisingly, these frontier models are often comparable or even worse than naive baselines on Cups, Coins, and Orderbook. Pattern learning also exhibits similar difficulty—many models barely outperform the naive baselines, especially on the forecasting-related datasets. Thus, these models still have room to improve.

LEAPS reveals orthogonal shortcomings of open-weight models. We show the results of the high-performing open-weight models at 128K input length in Figure 3; full results are in Figure 11. Although open-weight models are above 90% on recall tasks, they exhibit similar or worse performance to naive baselines on LEAPS, which suggests that models are still not able to effectively track states across long trajectories or learn patterns from many observations.

3.3 LEAPS SHOWS SIGNIFICANT DEGRADATION WITH LONGER LENGTH

We present the results from 8K to 128K input length for selected models in Figure 4, relative to each model's performance at 8K. The full results and absolute numbers are in Figure 11.

Variable degradation at long context. The three capabilities show distinct trends as the context plays fundamentally different roles in each: with longer inputs, recall results are relatively consistent, state tracking smoothly declines, whereas pattern learning observes some increase in performance, as strong models can make use of the additional data, before decreasing. The decrease from 8K to 128K over all models observes greater statistical significance on state tracking and pattern learning than recall. State tracking is especially difficult because it requires composing a long chain of transformations, and missing any step would sabotage the performance.

[8] We discard the samples where the API refuses to respond, which happens less than 4% of the time.

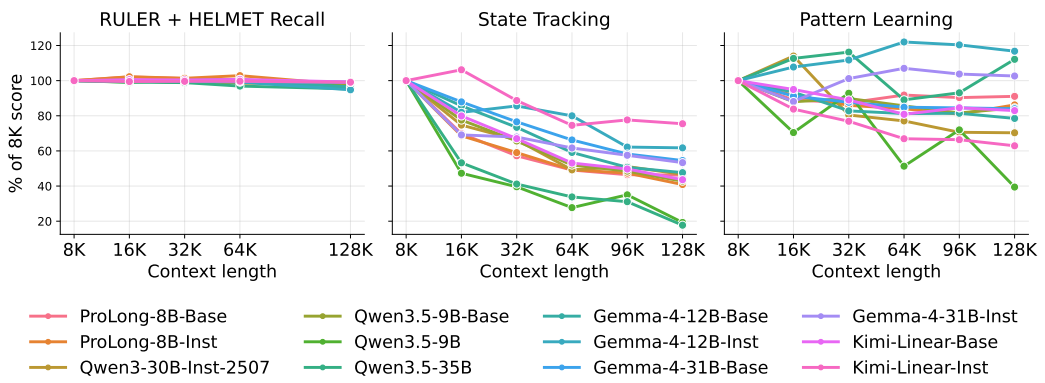

Figure 4: Performance is shown relative to each model's performance at 8K input tokens. Models observe substantially more degradation with increasing length on state tracking and pattern learning compared to recall. Model performance drops by up to 80% from 8K to 128K on LEAPS.

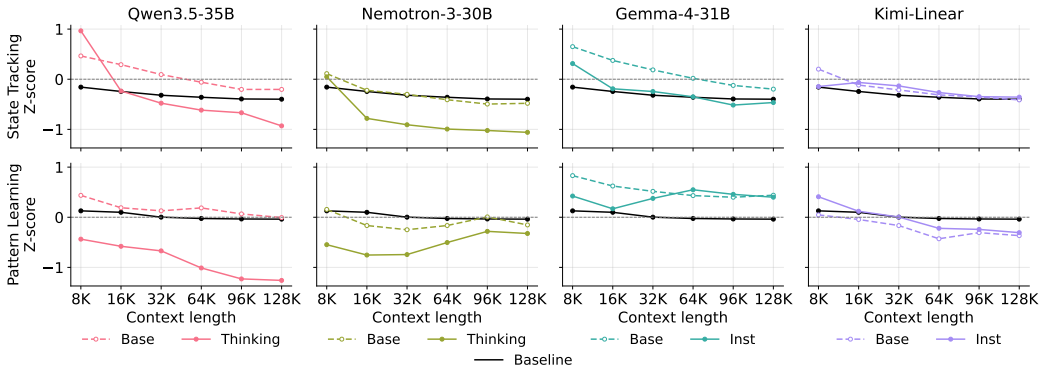

Figure 5: Comparison of base and post-trained models. Results shown in z-scores to standardize averaging across datasets and comparisons across input lengths.

Thinking models underperform their base counterparts. Across different model families of Qwen3.5, Nemotron, Gemma, and Kimi Linear, we find that the base models generally perform better or equal to their post-trained counterparts in Figure 5. To effectively assess overall performance across datasets and input lengths, we normalize the absolute performance into z-scores in each dataset before taking the average; details are in §A.2.

Thinking models, Qwen3.5-35B and Nemotron-3-30B, are substantially worse than their base models, by up to 1 standard deviation. Qualitatively, we find that existing thinking models often spend most of their generation budget on reasoning traces or make wrong assumptions; more in Section 4.3. In contrast, base models generate better answers by relying on their hidden representations. Instruction-tuning may result in slight degradation in performance at shorter input lengths, but Gemma-4 and Kimi-Linear achieve similar performance at 128K input length. Thus, how to post-train effective LCLMs to improve state tracking and pattern learning remains an open question.

4 ANALYSIS

4.1 LEAPS DATASETS SPAN DISTINCT CAPABILITIES

We plot the correlation between different datasets in LEAPS in Figure 7. Intuitively, many of the pattern learning tasks tend to correlate highly with each other—Chess, RPS, and Forecast exhibit significant positive correlation with each other. However, the market movement prediction tasks, Kaggle and Social Media, exhibit relatively low correlation with all other tasks, suggesting that predicting stock returns may test for different capabilities. Thus, the diverse range of datasets allows us to analyze LCLMs along multiple aspects of long-context capabilities.

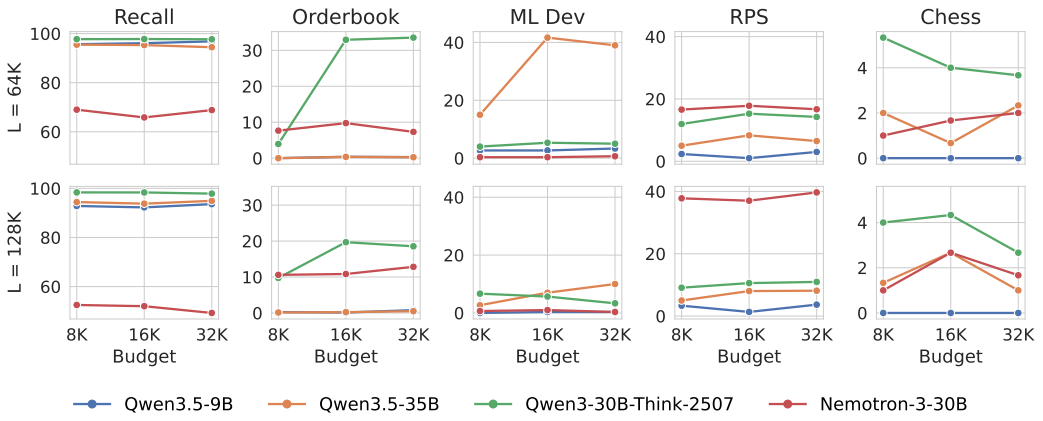

Figure 6: Results of thinking models with generation budgets from 8K to 32K, on input lengths $L = \{64\text{K}, 128\text{K}\}$. Performance may improve going from 8K to 16K output but plateaus after.

Furthermore, the state-tracking tasks show moderately strong correlation with each other, with the exception of the Cups dataset that shows low correlation with all other datasets. Surprisingly, most state-tracking datasets also correlate strongly with pattern learning tasks. Notably, ML Dev and Code Tracking tasks show medium to strong correlation with Social Media, Forecast, Ranking, Chess, and RPS. Thus, state tracking and pattern learning capabilities may be intertwined.

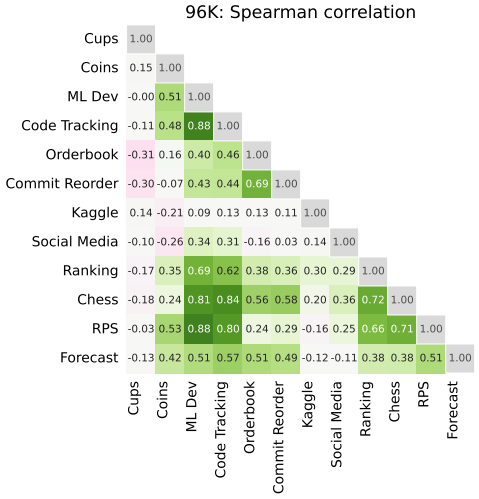

Figure 7: Correlation matrix at $L = 96\text{K}$, calculated across 23 models: API models and open-weight non-thinking models. The standard error of Spearman $\rho = 0$ is 0.209.

4.2 More thinking budget does not help

For our main experiments, we give the thinking models a budget of $B = 16\text{K}$ extra tokens during generation to try to ensure that generation budget was not a limiting factor; for reference, previous work set $B = 8\text{K}$ during long-context evaluation[9] (Yang et al., 2025). However, we observed that many thinking models, especially the Qwen models, do not output a final answer due to running out of generation budget on some tasks in Figure 13. While reasoning efficiently is *also* an important skill, in these benchmarks, we focused on a model's terminal ability to arrive at the correct answer, and thus ran an ablation where we varied the number of allowed generation tokens between 8K and 32K to test if additional thinking tokens led to better results. We show the results in Figure 6.

Recall observes little change with changing generation budget. On LEAPS, performance may increase as B increases from 8K to 16K tokens—suggesting that our tasks are more difficult and require more thinking. However, performance plateaus past $B = 16\text{K}$; some settings may even incur some degradation, such as Chess at 128K. Thus, more thinking is not enough for solving LEAPS.

4.3 Model behaviors

Many LCLMs perform similarly or even worse than naive baselines, which resembles random guess by minimally using the context; we further investigate the failure cases through qualitative analysis, as shown in Table 2. We observe two interesting behaviors—relying on brute-force methods in state tracking and making overly simple assumptions in pattern learning. Many models often manually step through all operations, exhausting their generation budget, suggesting a fundamental limitation of how existing LCLMs use their hidden representations for state tracking. We also find that existing

[9]Details on `https://huggingface.co/Qwen/Qwen3-30B-A3B-Thinking-2507`

Qwen 3.5 30B Cups reasoning	GPT-OSS-120B Kaggle reasoning
The user wants to know the final positions of three balls. ... I need to go through the list of "On table X, swap cups Y and Z" instructions and filter for those that apply to Table 58. 1. On table 58, swap cups 3 and 2. 2. On table 58, swap cups 3 and 4. ... 816. On table 58, swap cups 5 and 2. 817. On table 58, swap cups **[run out of generation budget]**	maybe the responder is simply the last value of "Continuous data"? ··· Take first example: ··· responder equals last continuous data value. Check second example: answer (C) responder 0.0412. Continuous data ends with "... 0.0412". Yes. So responder = last element of continuous data. The continuous data line ends with: "... -0.3101, 2.3888, 1.3782, 0.7434". Yes last value is 0.7434. Thus responder = 0.7434. Thus output: (E) **[Gold answer: (C)]**

Table 2: Examples of unsuccessful model reasoning at 128K input tokens. Qwen 3.5 30B exhausts its generation budget while explicitly tracking swaps in a Cups sample (left), while GPT-OSS-120B assumes a spurious pattern and returns an incorrect answer on a Kaggle sample (right).

models can also assume spurious patterns in pattern learning, leading to incorrect conclusions. For example, GPT-OSS-120B's reasoning chain incorrectly assumes the responder is just the last number in the numerical feature due to two in-context examples, and ignores the rest of the input. Additional analysis and ablations are in §A.3.

5 Related Works

Long-context benchmarks. Synthetic tasks have been popular in long-context evaluation (Hsieh et al., 2024; Kamradt, 2023; OpenAI, 2025; Vodrahalli et al., 2024). However, they are limited to retrieval settings, which do not cover other long-context capabilities. Recent benchmarks have also extended evaluations to question-answering, RAG, summarization, code analysis (Zhang et al., 2024; Yen et al., 2025; Bai et al., 2025; Lu et al., 2026; Lee et al., 2024). These tasks still primarily test retrieval-related capabilities, whereas state tracking and pattern learning capabilities required by agentic applications and behavioral simulations are still overlooked.

Many-shot in-context learning has also been used to test if models can reliably learn new tasks on the fly (Bertsch et al., 2025a; Li et al., 2025; Yen et al., 2025), but our pattern-learning tasks extend context learning to more complex settings beyond classification tasks. CLBench also tests the ability to learn from context (Dou et al., 2026b;a) and LongProc (Ye et al., 2025), but they are more instruction-focused and cannot be easily scaled to longer contexts. Aggregation tasks have also been studied (Bertsch et al., 2025b), but our state-tracking datasets cover more agentic usage of LCLMs. State tracking Toshniwal et al. (2022); Liu et al. (2023); Merrill et al. (2024) and forecasting (Tan et al., 2024; Gruver et al., 2023; Halawi et al., 2024) have been studied in short-context settings before, and we are the first to apply them in long-context settings.

Long-context language models. LCLMs power many powerful long-horizon agents, but the development of these models still evaluates on few synthetic tasks (GLM-5-Team et al., 2026; Team et al., 2026; 2025; Gao et al., 2025, *inter alia*). Furthermore, many behavioral studies leverage LLMs for social simulations (Park et al., 2023; Wang et al., 2023; Ziems et al., 2024), and LCLMs can be a powerful tool for ingesting and learning user behaviors. In this work, we develop LEAPS to further the progress of building more holistic and powerful LCLMs.

6 Conclusion

We introduce LEAPS for evaluating two critical yet understudied long-context capabilities—*state tracking* and *pattern learning*. LEAPS consists of 6 novel datasets per capability, including agentic coding and forecasting. Our benchmark includes reliable evaluation metrics, scales to millions of tokens, and is robust to contamination. Examining thirty models at context lengths from 8K to 512K tokens, we find that even frontier models struggle much more with state tracking and pattern learning as compared to more classical long-context recall-oriented tasks. Thus, these benchmarks are better able to distinguish model performance along these two new long-context axes, both of which continue to hold room for improvement, even for the strongest models. Finally, our analysis shows that existing models degrade significantly more with longer lengths and many thinking models struggle on LEAPS. We hope LEAPS can be a resource for building the next generation of LCLMs.

AI USE STATEMENT

In this work, we used generative AI tools to implement code for procedural generation of synthetic data, data/result visualization, and raw data pre-processing. We have not used generative AI tools for proposing/refining hypotheses, designing/providing feedback on research methodologies/experiments, supporting qualitative and thematic data analysis, or interpreting results. Developing theoretical models, formulating/proving mathematical claims, writing proofs, and assisting with translation are not applicable to this work. Additionally, we used generative AI tools for proofreading and grammatical assistance, but the initial draft and most subsequent revisions were entirely done by humans. We have reviewed all AI-assisted work. Specifically, LLM-generated code was verified by 2 authors and tested for correctness by at least one author. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We release the code and instructions for generating the data and evaluating the results. We also release the original evaluation data as Parquet files for ease of use upon publication, with the exception of the social media benchmark due to licensing restrictions. The evaluation and data generation code can be found in the supplemental materials. The data is excluded from the supplemental materials due to the limited size allowed by OpenReview. Specific descriptions of the constructions of the data and the experimental settings can be found in §A.1 and §A.2.

REFERENCES

Rishabh Agarwal, Avi Singh, Lei M Zhang, Bernd Bohnet, Luis Rosias, Stephanie C.Y. Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, John D Co-Reyes, Eric Chu, Feryal Behbahani, Aleksandra Faust, and Hugo Larochelle. Many-shot in-context learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=AB6XpMzvqH.

Anthropic. Claude Opus 4.8 system card. Anthropic, 2026a. URL https://www.anthropic.com/claude-opus-4-8-system-card.

Anthropic. Claude Sonnet 5 system card. Technical report, Anthropic, June 2026b. URL https://www.anthropic.com/claude-sonnet-5-system-card.

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Re. Zoology: Measuring and improving recall in efficient language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=LY3ukUANko.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3639–3664, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.183. URL https://aclanthology.org/2025.acl-long.183/.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

Amanda Bertsch, Maor Ivgi, Emily Xiao, Uri Alon, Jonathan Berant, Matthew R. Gormley, and Graham Neubig. In-context learning with long-context models: An in-depth exploration. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 12119–12149, Albuquerque, New Mexico, April 2025a. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.605. URL https://aclanthology.org/2025.naacl-long.605/.

Amanda Bertsch, Adithya Pratapa, Teruko Mitamura, Graham Neubig, and Matthew R. Gormley. Oolong: Evaluating long context reasoning and aggregation capabilities, 2025b. URL https://arxiv.org/abs/2511.02817.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Maanit Desai, Yirun Zhang, Ryan Holbrook, Kait O'Neil, and Maggie Demkin. Jane Street real-time market data forecasting. https://www.kaggle.com/competitions/jane-street-real-time-market-data-forecasting, 2024. Kaggle.

Shihan Dou, Yujiong Shen, Chenhao Huang, Junjie Ye, Jiayi Chen, Junzhe Wang, Qianyu He, Shichun Liu, Changze Lv, Jiahang Lin, Jiazheng Zhang, Ming Zhang, Shaofan Liu, Tao Ji, Zhangyue Yin, Cheng Zhang, Huaibing Xie, Jianglu Hu, Jingcheng Deng, Lincheng Li, Minda Hu, Shaolei Wang, Syrus Zhao, Weichao Wang, Yan Lei, Yang Liu, Yanling Xiao, Yiting Liu, Zenan Xu, Zhen Guo, Ziliang Zhao, Pluto Zhou, Tao Gui, Qi Zhang, Xuanjing Huang, Yu-Gang Jiang, Di Wang, and Shunyu Yao. Cl-bench life: Can language models learn from real-life context?, 2026a. URL https://arxiv.org/abs/2604.27043.

Shihan Dou, Ming Zhang, Zhangyue Yin, Chenhao Huang, Yujiong Shen, Junzhe Wang, Jiayi Chen, Yuchen Ni, Junjie Ye, Cheng Zhang, Huaibing Xie, Jianglu Hu, Shaolei Wang, Weichao Wang, Yanling Xiao, Yiting Liu, Zenan Xu, Zhen Guo, Pluto Zhou, Tao Gui, Zuxuan Wu, Xipeng Qiu, Qi Zhang, Xuanjing Huang, Yu-Gang Jiang, Di Wang, and Shunyu Yao. Cl-bench: A benchmark for context learning, 2026b. URL https://arxiv.org/abs/2602.03587.

Benjamin James Dyson, Jonathan Michael Paul Wilbiks, Raj Sandhu, Georgios Papanicolaou, and Jaimie Lintag. Negative outcomes evoke cyclic irrational decisions in rock, paper, scissors. *Scientific reports*, 6(1):20479, 2016.

Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. How to train long-context language models (effectively). In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7376–7399, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.366. URL https://aclanthology.org/2025.acl-long.366/.

Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=flNZJ2eOet.

GLM-5-Team, :, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, Jiajie Zhang, Jian Jiao, Jiaqi Guo, Jingsen Wang, Jingzhao Du, Jinzhu Wu, Kedong Wang, Lei Li, Lin Fan, Lucen Zhong, Mingdao Liu, Mingming Zhao, Pengfan Du, Qian Dong, Rui Lu, Shuang-Li, Shulin Cao, Song Liu, Ting Jiang, Xiaodong Chen, Xiaohan Zhang, Xuancheng Huang, Xuezhen Dong, Yabo Xu, Yao Wei, Yifan An, Yilin Niu, Yitong Zhu, Yuanhao Wen, Yukuo Cen, Yushi Bai, Zhongpei Qiao, Zihan Wang, Zikang Wang, Zilin Zhu, Ziqiang Liu, Zixuan Li, Bojie Wang, Bosi Wen, Can Huang, Changpeng Cai, Chao Yu, Chen Li, Chengwei Hu, Chenhui Zhang, Dan Zhang, Daoyan Lin, Dayong Yang, Di Wang, Ding Ai, Erle Zhu, Fangzhou Yi, Feiyu Chen, Guohong Wen, Hailong Sun, Haisha Zhao, Haiyi Hu, Hanchen Zhang, Hanrui Liu, Hanyu Zhang, Hao Peng, Hao Tai, Haobo Zhang, He Liu, Hongwei Wang, Hongxi Yan, Hongyu Ge, Huan Liu, Huanpeng Chu,

Jia'ni Zhao, Jiachen Wang, Jiajing Zhao, Jiamin Ren, Jiapeng Wang, Jiaxin Zhang, Jiayi Gui, Jiayue Zhao, Jijie Li, Jing An, Jing Li, Jingwei Yuan, Jinhua Du, Jinxin Liu, Junkai Zhi, Junwen Duan, Kaiyue Zhou, Kangjian Wei, Ke Wang, Keyun Luo, Laiqiang Zhang, Leigang Sha, Liang Xu, Lindong Wu, Lintao Ding, Lu Chen, Minghao Li, Nianyi Lin, Pan Ta, Qiang Zou, Rongjun Song, Ruiqi Yang, Shangqing Tu, Shangtong Yang, Shaoxiang Wu, Shengyan Zhang, Shijie Li, Shuang Li, Shuyi Fan, Wei Qin, Wei Tian, Weining Zhang, Wenbo Yu, Wenjie Liang, Xiang Kuang, Xiangmeng Cheng, Xiangyang Li, Xiaoquan Yan, Xiaowei Hu, Xiaoying Ling, Xing Fan, Xingye Xia, Xinyuan Zhang, Xinze Zhang, Xirui Pan, Xu Zou, Xunkai Zhang, Yadi Liu, Yandong Wu, Yanfu Li, Yidong Wang, Yifan Zhu, Yijun Tan, Yilin Zhou, Yiming Pan, Ying Zhang, Yinpei Su, Yipeng Geng, Yong Yan, Yonglin Tan, Yuean Bi, Yuhan Shen, Yuhao Yang, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yurong Wu, Yutao Zhang, Yuxi Duan, Yuxuan Zhang, Zezhen Liu, Zhengtao Jiang, Zhenhe Yan, Zheyu Zhang, Zhixiang Wei, Zhuo Chen, Zhuoer Feng, Zijun Yao, Ziwei Chai, Ziyuan Wang, Zuzhou Zhang, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Google DeepMind. Gemini 3.5 flash: Model card. Google DeepMind, May 2026. URL https://deepmind.google/models/model-cards/gemini-3-5-flash/.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria,

Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew G Wilson. Large language models are zero-shot time series forecasters. *Advances in Neural Information Processing Systems*, 36:19622–19635, 2023.

Danny Halawi, Fred Zhang, Chen Yueh-Han, and Jacob Steinhardt. Approaching human-level forecasting with language models. *Advances in Neural Information Processing Systems*, 37:50426–

50468, 2024.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

Garrett Kamradt. Needle in a haystack - pressure testing LLMs, 2023. URL https://github.com/gkamradt/LLMTest_NeedleInAHaystack.

Ezra Karger, Houtan Bastani, Chen Yueh-Han, Zachary Jacobs, Danny Halawi, Fred Zhang, and Philip E. Tetlock. Forecastbench: A dynamic benchmark of AI forecasting capabilities. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=lfPkGWXLLf.

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 Roa 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. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=CGNJL6CeV0.

Jinhyuk Lee, Anthony Chen, Zhuyun Dai, Dheeru Dua, Devendra Singh Sachan, Michael Boratko, Yi Luan, Sébastien M. R. Arnold, Vincent Perot, Siddharth Dalmia, Hexiang Hu, Xudong Lin, Panupong Pasupat, Aida Amini, Jeremy R. Cole, Sebastian Riedel, Iftekhar Naim, Ming-Wei Chang, and Kelvin Guu. Can long-context language models subsume retrieval, RAG, SQL, and more?, 2024. URL https://arxiv.org/abs/2406.13121.

Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. Long-context LLMs struggle with long in-context learning. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=Cw2xlg0e46.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=De4FYqjFueZ.

Zhiyuan Lu, Chenliang Li, Yingcheng Shi, Weizhou Shen, Ming Yan, and Fei Huang. Corpusqa: A 10 million token benchmark for corpus-level analysis and reasoning, 2026. URL https://arxiv.org/abs/2601.14952.

Xing Han Lù. BM25S: Orders of magnitude faster lexical search via eager sparse scoring, 2024. URL https://arxiv.org/abs/2407.03618.

William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545, 2023.

William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. In *Forty-first International Conference on Machine Learning*, 2024.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11048–11064, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.759. URL https://aclanthology.org/2022.emnlp-main.759/.

Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert, 2020. URL https://arxiv.org/abs/1901.04085.

NVIDIA, :, Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, Aditya Vavre, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, Aleksandr Shaposhnikov, Alex Kondratenko, Alexander Bukharin, Alexandre Milesi, Ali Taghibakhshi, Alisa Liu, Amelia Barton, Ameya Sunil Mahabaleshwarkar, Amir Klein, Amit Zuker, Amnon Geifman, Amy Shen, Anahita Bhiwandiwalla, Andrew Tao, Ann Guan, Anubhav Mandarwal, Arham Mehta, Ashwath Aithal, Ashwin Poojary, Asif Ahamed, Asma Kuriparambil Thekkumpate, Ayush Dattagupta, Banghua Zhu, Bardiya Sadeghi, Barnaby Simkin, Ben Lanir, Benedikt Schifferer, Besmira Nushi, Bilal Kartal, Bita Darvish Rouhani, Boris Ginsburg, Brandon Norick, Brandon Soubasis, Branislav Kisacanin, Brian Yu, Bryan Catanzaro, Carlo del Mundo, Chantal Hwang, Charles Wang, Cheng-Ping Hsieh, Chenghao Zhang, Chenhan Yu, Chetan Mungekar, Chintan Patel, Chris Alexiuk, Christopher Parisien, Collin Neale, Damon Mosk-Aoyama, Dan Su, Dane Corneil, Daniel Afrimi, Daniel Rohrer, Daniel Serebrenik, Daria Gitman, Daria Levy, Darko Stosic, David Mosallanezhad, Deepak Narayanan, Dhruv Nathawani, Dima Rekesh, Dina Yared, Divyanshu Kakwani, Dong Ahn, Duncan Riach, Dusan Stosic, Edgar Minasyan, Edward Lin, Eileen Long, Eileen Peters Long, Elena Lantz, Ellie Evans, Elliott Ning, Eric Chung, Eric Harper, Eric Tramel, Erick Galinkin, Erik Pounds, Evan Briones, Evelina Bakhturina, Faisal Ladhak, Fay Wang, Fei Jia, Felipe Soares, Feng Chen, Ferenc Galko, Frankie Siino, Gal Hubara Agam, Ganesh Ajjanagadde, Gantavya Bhatt, Gargi Prasad, George Armstrong, Gerald Shen, Gorkem Batmaz, Grigor Nalbandyan, Haifeng Qian, Harsh Sharma, Hayley Ross, Helen Ngo, Herman Sahota, Hexin Wang, Himanshu Soni, Hiren Upadhyay, Huizi Mao, Huy C Nguyen, Huy Q Nguyen, Iain Cunningham, Ido Shahaf, Igor Gitman, Ilya Loshchilov, Ivan Moshkov, Izzy Putterman, Jan Kautz, Jane Polak Scowcroft, Jared Casper, Jatin Mitra, Jeffrey Glick, Jenny Chen, Jesse Oliver, Jian Zhang, Jiaqi Zeng, Jie Lou, Jimmy Zhang, Jining Huang, Joey Conway, Joey Guman, John Kamalu, Johnny Greco, Jonathan Cohen, Joseph Jennings, Joyjit Daw, Julien Veron Vialard, Junkeun Yi, Jupinder Parmar, Kai Xu, Kan Zhu, Kari Briski, Katherine Cheung, Katherine Luna, Keshav Santhanam, Kevin Shih, Kezhi Kong, Khushi Bhardwaj, Krishna C. Puvvada, Krzysztof Pawelec, Kumar Anik, Lawrence McAfee, Laya Sleiman, Leon Derczynski, Li Ding, Lucas Liebenwein, Luis Vega, Maanu Grover, Maarten Van Segbroeck, Maer Rodrigues de Melo, Makesh Narsimhan Sreedhar, Manoj Kilaru, Maor Ashkenazi, Marc Romeijn, Mark Cai, Markus Kliegl, Maryam Moosaei, Matvei Novikov, Mehrzad Samadi, Melissa Corpuz, Mengru Wang, Meredith Price, Michael Boone, Michael Evans, Miguel Martinez, Mike Chrzanowski, Mohammad Shoeybi, Mostofa Patwary, Nabin Mulepati, Natalie Hereth, Nave Assaf, Negar Habibi, Neta Zmora, Netanel Haber, Nicola Sessions, Nidhi Bhatia, Nikhil Jukar, Nikki Pope, Nikolai Ludwig, Nima Tajbakhsh, Nirmal Juluru, Oleksii Hrinchuk, Oleksii Kuchaiev, Olivier Delalleau, Oluwatobi Olabiyi, Omer Ullman Argov, Ouye Xie, Parth Chadha, Pasha Shamis, Pavlo Molchanov, Pawel Morkisz, Peter Dykas, Peter Jin, Pinky Xu, Piotr Januszewski, Pranav Prashant Thombre, Prasoon Varshney, Pritam Gundecha, Qing Miao, Rabeeh Karimi Mahabadi, Ran El-Yaniv, Ran Zilberstein, Rasoul Shafipour, Rich Harang, Rick Izzo, Rima Shahbazyan, Rishabh Garg, Ritika Borkar, Ritu Gala, Riyad Islam, Roger Waleffe, Rohit Watve, Roi Koren, Ruoxi Zhang, Russell J. Hewett, Ryan Prenger, Ryan Timbrook, Sadegh Mahdavi, Sahil Modi, Samuel Kriman, Sanjay Kariyappa, Sanjeev Satheesh, Saori Kaji, Satish Pasumarthi, Sean Narentharen, Sean Narenthiran, Seonmyeong Bak, Sergey Kashirsky, Seth Poulos, Shahar Mor, Shanmugam Ramasamy, Shantanu Acharya, Shaona Ghosh, Sharath Turuvekere Sreenivas, Shelby Thomas, Shiqing Fan, Shreya Gopal, Shrimai Prabhumoye, Shubham Pachori, Shubham Toshniwal, Shuoyang Ding, Siddharth Singh, Simeng Sun, Smita Ithape, Somshubra Majumdar, Soumye Singhal, Stefania Alborghetti, Stephen Ge, Sugam Dipak Devare, Sumeet Kumar Barua, Suseella Panguluri, Suyog Gupta, Sweta Priyadarshi, Syeda Nahida Akter, Tan Bui, Teodor-Dumitru Ene, Terry Kong, Thanh Do, Tijmen Blankevoort, Tom Balough, Tomer Asida, Tomer Bar Natan, Tugrul Konuk, Twinkle Vashishth, Udi Karpas, Ushnish De, Vahid Noorozi, Vahid Noroozi, Venkat Srinivasan, Venmugil Elango, Vijay Korthikanti, Vitaly Kurin, Vitaly Lavrukhin, Wanli Jiang, Wasi Uddin Ahmad, Wei Du, Wei Ping, Wenfei Zhou, Will Jennings, William Zhang, Wojciech Prazuch, Xiaowei Ren, Yashaswi Karnati, Yejin Choi, Yev Meyer, Yi-Fu Wu, Yian Zhang, Ying Lin, Yonatan Geifman, Yonggan Fu, Yoshi Subara, Yoshi Suhara, Yubo Gao, Zach Moshe, Zhen Dong, Zihan Liu, Zijia Chen, and Zijie Yan. Nemotron 3 nano: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning, 2025. URL https://arxiv.org/abs/2512.20848.

OpenAI. Introducing GPT-4.1 in the API. https://openai.com/index/gpt-4-1/, 2025. Accessed: 2026-08-06.

OpenAI. GPT-5.6: Frontier intelligence that scales with your ambition. OpenAI, July 2026. URL https://openai.com/index/gpt-5-6/.

OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, Che Chang, Kai Chen, Mark Chen, Enoch Cheung, Aidan Clark, Dan Cook, Marat Dukhan, Casey Dvorak, Kevin Fives, Vlad Fomenko, Timur Garipov, Kristian Georgiev, Mia Glaese, Tarun Gogineni, Adam Goucher, Lukas Gross, Katia Gil Guzman, John Hallman, Jackie Hehir, Johannes Heidecke, Alec Helyar, Haitang Hu, Romain Huet, Jacob Huh, Saachi Jain, Zach Johnson, Chris Koch, Irina Kofman, Dominik Kundel, Jason Kwon, Volodymyr Kyrylov, Elaine Ya Le, Guillaume Leclerc, James Park Lennon, Scott Lessans, Mario Lezcano-Casado, Yuanzhi Li, Zhuohan Li, Ji Lin, Jordan Liss, Lily, Liu, Jiancheng Liu, Kevin Lu, Chris Lu, Zoran Martinovic, Lindsay McCallum, Josh McGrath, Scott McKinney, Aidan McLaughlin, Song Mei, Steve Mostovoy, Tong Mu, Gideon Myles, Alexander Neitz, Alex Nichol, Jakub Pachocki, Alex Paino, Dana Palmie, Ashley Pantuliano, Giambattista Parascandolo, Jongsoo Park, Leher Pathak, Carolina Paz, Ludovic Peran, Dmitry Pimenov, Michelle Pokrass, Elizabeth Proehl, Huida Qiu, Gaby Raila, Filippo Raso, Hongyu Ren, Kimmy Richardson, David Robinson, Bob Rotsted, Hadi Salman, Suvansh Sanjeev, Max Schwarzer, D. Sculley, Harshit Sikchi, Kendal Simon, Karan Singhal, Yang Song, Dane Stuckey, Zhiqing Sun, Philippe Tillet, Sam Toizer, Foivos Tsimpourlas, Nikhil Vyas, Eric Wallace, Xin Wang, Miles Wang, Olivia Watkins, Kevin Weil, Amy Wendling, Kevin Whinnery, Cedric Whitney, Hannah Wong, Lin Yang, Yu Yang, Michihiro Yasunaga, Kristen Ying, Wojciech Zaremba, Wenting Zhan, Cyril Zhang, Brian Zhang, Eddie Zhang, and Shengjia Zhao. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701320. doi: 10.1145/3586183.3606763. URL https://doi.org/10.1145/3586183.3606763.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, April 2009. ISSN 1554-0669. doi: 10.1561/1500000019. URL https://doi.org/10.1561/1500000019.

Mingtian Tan, Mike A. Merrill, Vinayak Gupta, Tim Althoff, and Thomas Hartvigsen. Are language models actually useful for time series forecasting? In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 60162–60191. Curran Associates, Inc., 2024. doi: 10.52202/079017-1922. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/6ed5bf446f59e2c6646d23058c86424b-Paper-Conference.pdf.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, Mayank Chaturvedi, Aditya Chawla, Victor Cotruta, Alice Coucke, Phil Culliton, Robert Dadashi, Lucas Dixon, Mohamed Elhawaty, Utku Evci, Clément Farabet, Johan Ferret, Filippo Galgani, Sertan Girgin, Jean-Bastien Grill, Maarten Grootendorst, Jiaxian Guo, Cassidy Hardin, Yanzhang He, Steven M. Hernandez, Omri Homburger, Léonard Hussenot, Juyeong Ji, Armand Joulin, Aishwarya Kamath, Parnian Kassraie, Olivier Lacombe, Preethi Lahoti, Gaël Liu, Gus Martins, Luciano Martins, Tatiana Matejovicova, Ramona Merhej, Nikola Momchev, Sneha Mondal, Ryan Mullins, Sindhu Raghuram Panyam, Shreya Pathak, Sarah Perrin, André Susano Pinto, Etienne Pot, Angéline Pouget, Alexandre Ramé, Sabela Ramos, Douglas Reid, David Rim, Morgane Rivière, Karsten Roth, Louis Rouillard, Omar Sanseviero, Pier Giuseppe Sessa, Shane Settle, Danila Sinopalnikov, Sara Smoot, Piotr Stanczyk, Andreas Steiner, Lawrence Stewart, Ilya Tolstikhin, Michael Tschannen, Anton Tsitsulin, Nino Vieillard, Renjie Wu, Pingmei Xu, Haichuan Yang, Edouard Yvinec, Biao Zhang, Li Zhang, Joe Zou, Nicolas Aagnes, Abdelrahman Abdelhamed, Jakub Adamek, Shivani Agrawal, Shubham Agrawal, Ibrahim Alabdulmohsin, Jean Baptiste Alayrac, Uri Alon, Chandramouli Amarnath, Ankesh Anand, Chrysovalantis Anastasiou, Setareh Ariafar, François-Xavier Aubet, Kyriakos Axiotis, Federico Barbero, Joelle Barral, Alexei Bendebury, Urs Bergmann, Stanley Bileschi, Kat Black, Mathieu Blondel, Sebastian Borgeaud, Arthur Bražinskas, Ryan Burnell, Robert Busa-Fekete, Mu Cai, Daniele Calandriello, Glenn Cameron, Charlotte Caucheteux,

Rahma Chaabouni, Garima Chadha, Jetha Chan, Blake Jianhang Chen, Jesse Chen, Lin Chen, Xu Chen, Derek Cheng, Tzu hsiang Chien, Nikolai Chinaev, Yi Chou, Zhaohui Chu, Benjamin Coleman, Pooja Consul, Sam Conway-Rahman, Scott Crowell, Dylan Cutler, Vivek Dani, Samira Daruki, Anil Das, Daniel Deutsch, Nishanth Dikkala, Li Ding, Qiuhan Ding, Shenil Dodhia, Konstantin Donhauser, Tulsee Doshi, Anca Dragan, Alex Druinsky, Sahil Dua, Zoltan Egyed, Danielle Eisenbud, Daniel Eppens, Cindy Fan, Bahare Fatemi, Yassir Fathullah, Vlad Feinberg, Milen Ferev, Sebastian Flennerhag, Takumi Fujimoto, João Gabriel Oliveira, Isaac Galatzer-Levy, João Gante, Simon Geisler, Soham Ghosal, Antonious M. Girgis, Tamara von Glehn, Alec Go, Alhaad Gokhale, Alex Grills, Yiming Gu, Mayank Gupta, Pramod Gupta, Guru Guruganesh, Raia Hadsell, Hamza Harkous, Jitendra Harlalka, Demis Hassabis, Anja Hauth, Joe Heyward, Arian Hosseini, Chih-Yang Hsia, I-Hung Hsu, Xiaopeng Huang, Yangsibo Huang, Kevin Hui, Adrian Hutter, Te I, Fotis Iliopoulos, Advait Jain, Ganesh Jawahar, Ziwei Ji, Qilin Jin, Melvin Johnson, Kandarp Joshi, Arun Kandoor, Wang-Cheng Kang, Koray Kavukcuoglu, Mehran Kazemi, Kathleen Kenealy, Amr Khalifa, Phoebe Kirk, Ivan Korotkov, Suraj Kothawade, Vitaly Kovalev, Neel Kovelamudi, Adam Kraft, Ravin Kumar, Vivek Kumar, Harish Kuppam, Justin Lannin, Chen-Yu Lee, Seungji Lee, Dmitry Lepikhin, Alon Levkovitch, Dongdong Li, Qiujia Li, Valentin Liévin, Ethan Lin, Ziqian Lin, Casper Liu, Tianlin Liu, Tianqi Liu, Xin Liu, Ivan Lobov, Mayank Lunayach, Min Ma, Gagan Madan, Andrii Maksai, Eric Malmi, Michal Matuszak, Daniel McDuff, Gaurav Menghani, Maciej Mikuła, Daniil Mirylenka, Karolis Misiunas, Vedant Misra, Andreea Mitran, Kareem Mohamed, Maksim Mukha, Eric Noland, James O'Donnell, Brendan O'Donoghue, Kate Olszewska, Bernett Orlando, Wanqiong Pan, Rina Panigrahy, Unnati Parekh, Nicolas Perez-Nieves, Chunjong Park, Eric Paskie, Liqian Peng, Bryce Petrini, Slav Petrov, Jonas Pfeiffer, Bilal Piot, Martyna Plomecka, Siim Poder, Octavio Ponce, Arijit Pramanik, David Racz, Anish Rajan, Michelle Ramanovich, Anand Rao, Marvin Ritter, Vitor Rodrigues, Evan Rosen, Mikołaj Rybiński, Noveen Sachdeva, Michaël E. Sander, Rohit Sathyanarayana, Sagar Savla, Samuel Schmidgall, Tal Schuster, George Scrivener, Benoit Seguin, Andrew Sellergren, Aliaksei Severyn, Izhak Shafran, Dhruv Shah, Bobak Shahriari, Yuan Shangguan, Ashish Shenoy, Pradeep Shenoy, Rakesh Shivanna, Pauline Sho, Lucas Spangher, Wojciech Stokowiec, Tim Strother, Yao Su, Yinghao Sun, Mukund Sundararajan, Andrea Tacchetti, Mor Hazan Taege, Pouya Tafti, Jean Tarbouriech, Chetan Tekur, Shantanu Thakoor, Rahul Thapa, Madeleine Traverse, Lenart Treven, Tao Tu, Chien Te Tung, Çağlar Ünlü, Petar Veličković, Malini Pooni Venkat, Sagar Gubbi Venkatesh, Vidya Venkiteswaran, Francesco Visin, Alex Vitvitskyi, Kiran Vodrahalli, Weiyi Wang, Xin Wang, Tris Warkentin, Jan Wassenberg, John Wieting, Cindy Wu, Lechao Xiao, Hao Xu, Yuhui Xu, Fuzhao Xue, Arun Yadav, Jun Yan, Antoine Yang, Lin Yang, Ming-Hsuan Yang, Ziyu Ying, Jae Hyeon Yoo, Morteza Zadimoghaddam, Sajjad Zafar, Fred Zhang, Jiageng Zhang, Jianyi Zhang, Xiaofan Zhang, Chao Zhao, David Zhou, and Chen Zou. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, Wentao Li, Enzhe Lu, Weizhou Liu, Yanru Chen, Weixin Xu, Longhui Yu, Yejie Wang, Yu Fan, Longguang Zhong, Enming Yuan, Dehao Zhang, Yizhi Zhang, T. Y. Liu, Haiming Wang, Shengjun Fang, Weiran He, Shaowei Liu, Yiwei Li, Jianlin Su, Jiezhong Qiu, Bo Pang, Junjie Yan, Zhejun Jiang, Weixiao Huang, Bohong Yin, Jiacheng You, Chu Wei, Zhengtao Wang, Chao Hong, Yutian Chen, Guanduo Chen, Yucheng Wang, Huabin Zheng, Feng Wang, Yibo Liu, Mengnan Dong, Zheng Zhang, Siyuan Pan, Wenhao Wu, Yuhao Wu, Longyu Guan, Jiawen Tao, Guohong Fu, Xinran Xu, Yuzhi Wang, Guokun Lai, Yuxin Wu, Xinyu Zhou, Zhilin Yang, and Yulun Du. Kimi linear: An expressive, efficient attention architecture, 2025. URL https://arxiv.org/abs/2510.26692.

Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Shubham Toshniwal, Sam Wiseman, Karen Livescu, and Kevin Gimpel. Chess as a testbed for language model state tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 11385–11393, 2022.

Kiran Vodrahalli, Santiago Ontanon, Nilesh Tripuraneni, Kelvin Xu, Sanil Jain, Rakesh Shivanna, Jeffrey Hui, Nishanth Dikkala, Mehran Kazemi, Bahare Fatemi, Rohan Anil, Ethan Dyer, Siamak Shakeri, Roopali Vij, Harsh Mehta, Vinay Ramasesh, Quoc Le, Ed Chi, Yifeng Lu, Orhan Firat, Angeliki Lazaridou, Jean-Baptiste Lespiau, Nithya Attaluri, and Kate Olszewska. Michelangelo:

Long context evaluations beyond haystacks via latent structure queries, 2024. URL https://arxiv.org/abs/2409.12640.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Daniel Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. Openhands: An open platform for AI software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=OJd3ayDDoF.

Zhijian Wang, Bin Xu, and Hai-Jun Zhou. Social cycling and conditional responses in the rock-paper-scissors game. *Scientific reports*, 4(1):5830, 2014.

Zhilin Wang, Yu Ying Chiu, and Yu Cheung Chiu. Humanoid agents: Platform for simulating human-like generative agents. In Yansong Feng and Els Lefever (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 167–176, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-demo.15. URL https://aclanthology.org/2023.emnlp-demo.15/.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=RdJVFCHjUMI.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=mXpq6ut8J3.

Xi Ye, Fangcong Yin, Yinghui He, Joie Zhang, Howard Yen, Tianyu Gao, Greg Durrett, and Danqi Chen. LongProc: Benchmarking long-context language models on long procedural generation. 2025.

Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. Helmet: How to evaluate long-context language models effectively and thoroughly. In *International Conference on Learning Representations (ICLR)*, 2025.

Hanshu Zhang, Frederic Moisan, and Cleotilde Gonzalez. Rock-paper-scissors play: beyond the win-stay/lose-change strategy. *Games*, 12(3):52, 2021.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. ∞Bench: Extending long context evaluation beyond 100K tokens. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15262–15277, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.acl-long.814.

Caleb Ziems, William Held, Omar Shaikh, Jiaao Chen, Zhehao Zhang, and Diyi Yang. Can large language models transform computational social science? *Computational Linguistics*, 50(1):237–291, March 2024. doi: 10.1162/coli_a_00502. URL https://aclanthology.org/2024.cl-1.8/.

A APPENDIX

A.1 DATASET CONSTRUCTION DETAILS

Cup swapping. This dataset is based on the classic cup swapping game, where balls are placed under cups, and the task is to keep track of which cup each ball is under. There are k tables, and each table has n cups and m balls, where $m < n$. Each ball is placed under exactly one cup, and we simulate random swap operations by randomly sampling tables, and pairs of cups in that table to swap. Humans can often do this task perfectly with minimal assistance, usually just a pen and paper are sufficient for tracking a large number of balls due to the simplicity of the task. This task tests the capacity of model state representation. In our main setting, we use $k = 100$, $n = 5$, and $m = 3$, and only query for the positions of balls on one table. That is, we only ask for the positions of the three balls on one particular table. We sample increasingly more operations to increase the input length.

For evaluation, we report the F1 score between the generated positions and the gold positions, and the baseline is random positions for the queried balls. We chose this as the baseline because a person who has lost track of the position can only perform their best guesses by randomly choosing the positions. In practice, we found that existing models always outputted exactly the queried balls in the output, which means the F1 score is practically recall.

Coins. Similar to cups, there are k tables with n piles of coins each. The key difference is that this task also includes a different type of operation: linear aggregation on each pile. Specifically, at each time step, we randomly sample an action, which can either swap two piles or add/remove a coin from one pile. We limit the number of coins in each pile to between 0 and 100—if a pile already has 100 coins, it cannot be sampled to add another coin.

During development, we had initially set the probability of adding and removing a coin to be equal for all piles. That is, for any given pile, the probabilities for adding, removing, and swapping are $45\%, 45\%, 10\%$, respectively. As a result, on expectation, the absolute number of coins in each pile stays the same, and given the relatively low chance of swapping, a very good heuristic was simply guessing the initial state of each pile. However, this doesn't require the model to perform much state tracking—guessing that there are roughly the same number of adds and removes is not really a long-context capability. Thus, for each pile, we use a beta distribution $p \sim \mathrm{Beta}(5, 5)$ to determine the probability of adding a coin for that pile (versus removal), which increases the variance of changes in order to thwart the trivial heuristic.

In our final setting, we use $k = 100$ and $n = 5$, and we only ask about 3 piles in evaluation. Given that the number of coins is limited between 0 and 100, we can normalize the L1 distance of the prediction by dividing by 100. Then, we report the average of the three normalized L1 distances, subtracted from 1, as the metric. The naive baseline is outputting the initial state of each pile, because that one might just guess no change after seeing roughly the same number of additions and subtractions[10].

Orderbook. We simulate a central limit orderbook to test if models can keep track of a highly dynamic environment. Specifically, we synthetically generate 10 actors who all trade in one market. Each actor has an internal fair value (or just "fair"), which is how much the actor values the security and is initialized by a Gaussian with mean 100 (an arbitrary choice). Then, at each step, we randomly sample an actor to take an action of buy, sell, or cancel.

When buying or selling, the actor samples random noise to add to its internal fair and to size its order. After an order is sent, it can result in one or more trades if there are matching resting orders (which cannot be from the same actor by construction). After a trade, both actors update their internal fairs by decreasing the fair if they just bought or increasing the fair if they just sold. After their fair is updated, each actor cancels any existing orders of theirs that would buy at a price higher than their fair or sell at a price below it.

[10] Another naive baseline could be guessing 50 (the midpoint) coins in each pile, but the instruction does not say that there can be at most 100 coins in each pile, and it may be an unreasonable guess if there were 1 coin in each pile at the start, for example.

The input context consists of two demonstrations of this task and just the orders; no transactions are shown. We simulate this until the length has reached the target number of tokens, and pose a question: If a particular new order were to arrive, how many shares would trade, and at what average price? This requires the model to keep track of existing orders (incorporating the effect of any cancels or trades that have taken place).

For the metrics, we report 1 minus the average between the normalized L1 distances for both the quantity and the average price. The quantity is normalized by dividing by the quantity in the query, as the number traded cannot exceed the latter. The guessed price is normalized by first clamping the value between the query price and the most aggressive price on the other side; for example, for a buy order, we would clamp to the window bounded by the lowest resting sell order price and by the query price. Then, we divide the L1 distance by the size of this window. If the number traded is 0, then the model is only rewarded with a score of 1 if both the output query and quantity are 0, otherwise the score is just 0. The naive baseline is just the query price and quantity. If one has lost track of the state, the only anchor point they would have is the query price and quantity.

ML Development. We simulate an ML model development environment, where the model has to follow specific instructions and steps to accomplish the model development goals outlined by the user. Specifically, the user begins with an instruction on how many GPUs to use during training and evaluation, which are sampled randomly, and all subsequent commands must follow that. Then, the model is shown multiple goals, each of which consists of a model size (6 possibilities), a training corpus (6 possibilities), a grid of learning rates and batch sizes (randomly sampled floats), a list of benchmarks to evaluate on (5 possibilities), and a production target to push to (randomly sampled uuid). The number of goals and the size of the learning rate and batch size grid increases with the target input length.

Then, to accomplish each goal, the model must issue the following commands, in order with the right arguments: train the model on the correct configuration (each one is run separately), inspect the train loss curves, evaluate on the designated downstream tasks, print a table summarizing the results for a set of models (only valid after all models have been evaluated), and push the best model according to the benchmark to production. In this environment, we can keep track of exactly all valid commands at any given step, and each command is mapped to exactly one goal that it advances.

The input also includes two demonstrations of example trajectories, both of which are relatively short. Finally, we show the model a history of the commands that have already been run and ask it to output another command that advances a specific goal. We report the metric of EM to any of the valid commands, and there is no naive baseline here, since the number of possible commands is infinite.

Chess. We first download games from the Lichess database[11] from the months of January 2026 to May 2026. We filter for all players who played at least 100 games and at least one at a rating of 2800 in any mode in May of 2026. We also filter out all bots (tagged by the website, checkable via the provided API call). Generating new games can be done by downloading the latest games.

We include metadata in the prompt: the date and time, time control, event (if applicable), and username and ELO of each player. For the moves, we show the progression of each game with the standard chess algebraic notation with the time remaining after each move.

For each player, we keep track of all chess states they have encountered throughout the first five months of 2026, and only evaluate on the first time they encounter a board state.

In the prompt, we show two demonstrations, each with two games in the history and one game to be formatted in the unfinished state. Then, the rest of the context is filled with the consecutive games that the player has played leading up to the test game. The test game has to last at least 20 turns, and we only ask questions about a turn at or after the 20th turn, so that it is less likely for the move to be part of a standard opening line. Notably, the board is in a state that the player has not encountered in any of the demonstration games, which means that the model cannot simply retrieve the move that the player has played at that state before.

[11] `https://database.lichess.org/`

We report the exact score with the actual move played as the metric. The naive baseline is outputting a random legal move at that state, since a person who is asked to play a move at any board state might just choose a random move. Although it would have been interesting to explore how the model moves differ from chess engine's moves, specific chess analysis is out of the scope for this work and we leave it for future work.

Code tracking and commit reordering. We source the commits from a list of permissively licensed repos for both code tracking and commit reordering. We select repos with active commit history in 2026 to avoid contamination with pre-training data of LLMs. The full list of all the Github repos is in Table 3. For both tasks, we also put a simple toy example of the task at the beginning of the prompt as in-context learning demonstration.

Code tracking. We first process all commits in 2026 and track metadata like which files each commit changes and the diffs from each commit. Then, we create an anchor for each month from January to May, which is that month's first mainline commit, which will be used as the starting point for all test cases. We also filter out patches that only edit binaries, lock files, and other non-code or large individual files over 100KB. Then, for a particular target input length, we first reserve the first 20% of the context window for showing the initial state of the files, and the remaining are for the commit messages and diffs. We pack the remaining context window with consecutive commits from the first day of the month, and then choose the initial file to display based on which ones were modified.

Then, we will choose a range of file lines to ask questions about. We first look at the git diff between the starting and ending states. We reject the line ranges where the markdown fences are included to make parsing later easier. For each starting anchor, we ask at most five questions about different sections of the files. We report the exact match between the output and the answer. The baseline is to guess the original state of the code in the file, but by construction that is 0 because we only query the lines that have been modified.

Commit reordering. We don't need to use the anchor at the start of every month as we don't need to show the initial state of each file. We build sequences of commits starting from the first day of 2026 and break them into chunks of the target lengths. Windows with fewer than four commits are discarded. We also filter out diff patches on binary files, lock files, and other oversized files. We render each commit with their commit message and diffs and remove the git blob hashes. We then find all consecutive pairs of the commit messages with dependencies—pairs where the second commit modifies (or removes) a line that was added by the first commit. Finally, we randomly swap one such pair and ask the model to identify it. We report the exact match score as the metric, and the naive baseline is outputting a random pair, which is $\frac{1}{n-1}$ on expectation, where n is the number of commits.

Rock paper scissors. Humans often exhibit predictable strategies in RPS (Dyson et al., 2016; Zhang et al., 2021; Wang et al., 2014). We simulate players with the strategies below:

1. Fixed random: probabilities are randomly sampled and fixed regardless of the opponent's moves. Note that this is not how humans play this game but we add this for variety.
2. Win-Stay/Lose-Change (WSLC): win leads to increase in the probability of playing the same move by Δp, lose can lead to either change 50/50, upgrade (change to the move that would have beat the move it just played), or downgrade (change to the move that would have lost to the prev move).
3. Win-Change/Lose-Stay (WCLS): the opposite of WSLC.
4. Adversarial: given the opponent's most popular move in the past m turns, play the move that beats that move. Evenly split the probability if there are multiple most played moves by the opponents.

Each player's strategy is randomly chosen and their parameters, such as Δp and m, are also randomly initialized. There are an infinite number of possible players and game histories, but we always have a reliable latent model of each player's distribution at any point. We show the model a history of games played between two players, and we ask the model to output the probability distribution over each player's next moves.

Repo	URL	License
Fabric	https://github.com/danielmiessler/Fabric	MIT
PowerShell	https://github.com/PowerShell/PowerShell	MIT
Vibe-Trading	https://github.com/HKUDS/Vibe-Trading	MIT
agency-agents	https://github.com/msitarzewski/agency-agents	MIT
astropy	https://github.com/astropy/astropy	BSD-3-Clause
biome	https://github.com/biomejs/biome	MIT/Apache-2.0
codebase-memory-mcp	https://github.com/DeusData/codebase-memory-mcp	MIT
curl	https://github.com/curl/curl	curl
django	https://github.com/django/django	BSD-3-Clause
flash-attention	https://github.com/Dao-AILab/flash-attention	BSD-3-Clause
ghostty	https://github.com/ghostty-org/ghostty	MIT
gin	https://github.com/gin-gonic/gin	MIT
json	https://github.com/nlohmann/json	MIT
llama.cpp	https://github.com/ggml-org/llama.cpp	MIT
maigret	https://github.com/soxoj/maigret	MIT
memos	https://github.com/usememos/memos	MIT
mlx	https://github.com/ml-explore/mlx	MIT
narwhals	https://github.com/narwhals-dev/narwhals	MIT
no-mistakes	https://github.com/kunchenguid/no-mistakes	MIT
numpy	https://github.com/numpy/numpy	BSD-3-Clause
ollama	https://github.com/ollama/ollama	MIT
openclaw	https://github.com/openclaw/openclaw	MIT
openpilot	https://github.com/commaai/openpilot	MIT
oxidizer	https://github.com/microsoft/oxidizer	MIT
pixi	https://github.com/prefix-dev/pixi	BSD-3-Clause
polars	https://github.com/pola-rs/polars	MIT
pytest	https://github.com/pytest-dev/pytest	MIT
rattler	https://github.com/conda/rattler	BSD-3-Clause
ripgrep	https://github.com/BurntSushi/ripgrep	Unlicense/MIT
ruff	https://github.com/astral-sh/ruff	MIT
scikit-learn	https://github.com/scikit-learn/scikit-learn	BSD-3-Clause
semantic-kernel	https://github.com/microsoft/semantic-kernel	MIT
sympy	https://github.com/sympy/sympy	BSD-3-Clause
torchtitan	https://github.com/pytorch/torchtitan	BSD-3-Clause
ty	https://github.com/astral-sh/ty	MIT
uv	https://github.com/astral-sh/uv	Apache-2.0/MIT
vision	https://github.com/pytorch/vision	BSD-3-Clause
zstd	https://github.com/facebook/zstd	BSD-3-Clause

Table 3: Repositories and licenses used for the code tracking and commit reordering datasets.

The metric is one minus the average Hellinger distance, and the baseline is always guessing $\frac{1}{3}$ for all moves, because it's a common prior that people would have equal preference over three random moves.

Review ranking. We simulate an Area Chair who has to rank a list of papers. The area chair has a hidden preference over five axes: Novelty, Significance, Soundness, Presentation, and Reproducibility. The hidden preference $\vec{w}$ consists of five uniformly sampled weights between $[0, 1]$.

Each paper p consists of five reviews, each of which can be mapped to a vector of scores $\vec{r}_{p,i}$ over the five axes. The score for each axis is an integer from 1 to 5, inclusive. We determine the scores for each review in the generation process, where a hidden Markov model (HMM) generates a set of pre-defined phrases given a list of scores. That is, each score on one axis maps to a list of eight pre-defined phrases, and the HMM emits one of them at random given a score. The entire review is a random shuffle order of these phrases. Please refer to the code base for the mapping from numerical values to the specific phrases.

Then, the overall score vector for a paper is simply the average $\vec{r}_p = \frac{1}{n}\sum_{i=1}^{n} \vec{r}_{p,i}$, and the AC's preference score for the paper is $\vec{r}_p \cdot \vec{w}$. The AC's preference over the given paper is sorted by their preference scores, and the model needs to learn that preference from the demonstrations and apply it to the new set of test papers. The number of test papers increases as the input length increases.

The metric is NDCG@20, and the baseline is returning the same order in which the papers appeared. This should be equal to random guess in expectation because all papers are generated randomly and the scores have no dependency on the ordering.

Kaggle. We use a time-series market forecasting Kaggle competition for testing if models can learn relationships between abstract features and market movements (Desai et al., 2024).[12] Both the data and the features are anonymized, which means that there is no identifying information in the text that might link any specific examples to a ticker. Furthermore, the fine granularity of the time interval means that it is highly unlikely for the specific timestamp to have appeared in the training set of any LLMs. Regardless, it should be relatively easy to re-generate this dataset with fresh data by collecting market data.

In this task, the input is a long history of demonstrations of the tuple features and a responder (i.e., the market movement in the next 30 minutes). The test instance only shows the features and the model must predict the responder. All demonstrations are the tuples for the same ticker in the timeframe immediately preceding the test instance, and the number of demonstrations is dependent on the target input length.

In our preliminary experiments, we asked the model to directly output the responder, which could be a float with many decimal points, and calculate distance as the metric. However, this was difficult to interpret and it was unclear how to normalize the distances between 0 and 1 for the sake of averaging with other datasets. Furthermore, models often performed pretty poorly when prompted to output a number, especially some weaker models. Thus, we formulated it as a multiple-choice question-answering task, with five approximately equally weighted bins, such that picking any fixed choice is equal to random guessing in expectation.

The metric is exact match (EM), and the baseline is always picking (C), which corresponds to no movements and is equal to random guessing in expectation.

Social Media Market Forecasting. Forecasting is a powerful application of LCLMs, and we also test the ability to predict market movements from textual features using an internal dataset. While we cannot release the underlying data, in this task we match content from social media platforms with tradable assets and measure our ability to predict the market movement of an asset in the time interval corresponding to the associated collection of relevant social media posts from that period. In our dataset, our test rows are given metadata and textual features, and we fill in context with demonstrations of feature responder pairs from previous timestamps. Similarly to Kaggle, we also formulate this task as an MCQA task with five bins. The metric is also EM and the naive baseline is always choosing (C), which is the same as random guess in expectation. A natural guess for

[12] www.kaggle.com/competitions/jane-street-real-time-market-data-forecasting

predicting market movements is simply guessing no movement, and this results in roughly 20% in EM.

Forecast. We use ForecastBench (Karger et al., 2025) as the source of forecast queries. These queries span many domains, including prediction markets (e.g., Kalshi, Manifold) and general world events (e.g., Wikipedia, the FED). We use the data from the query sets released for May 7th and May 24th of 2026. These queries are resolved a week later, which are May 14th and May 31st, respectively. In general, the articles are crawled on the date that they are published, even though CC News doesn't guarantee this. Refer to the original work for additional details on the data.

The contribution of our work is augmenting each query with news articles from CC News, which consists of a list of crawled articles for every day from curated sources.[13] Specifically, we first extract and process all news articles from May 1st to May 24th. Note that if an article was crawled on May 24th, it means that it was published then by the latest, which means that it could not have any information on what actually happens later—the answer/resolution cannot exist in any of the articles, and the model must infer the answer from the available information. We build a retrieval corpus for each day of the news using BM25S (Lù, 2024), an implementation of BM25 (Robertson & Zaragoza, 2009). For each query, we retrieve the top k articles from each day. Both k and the number of days we include preceding the query date scale with respect to the input length we are targeting.

We use the queries from May 24th as the test instances and the May 7th set as demonstrations for the model to learn from. We use 1 minus Brier score as the metric, and the naive baseline is the market value at the freeze date, which comes before the query date, and 50% if the market value is not available (i.e., the task is from a source that is not a prediction market).

A.2 Evaluation details

For the API models, we evaluate 50 samples on each dataset due to cost, and for all other models, we evaluate on 300 samples per dataset. For all evaluations, we use VLLM version 0.23.0 and Transformers version 5.12.1. The generation token is tuned for each task such that it is always possible for the model to generate the correct answer within the generation budget. For base models, we directly complete the input prompt and generate the final answer. For instruction-tuned models, we use model-specific chat templates and add 256 tokens to the generation budget to account for overhead of assistant messages. For thinking models, we use the model-specific chat template and reasoning parser, and add 16,384 tokens to the generation budget to enable longer chains of thought. Note that this often implies that many models will degenerate at 128K input length as the total tokens between input and output will exceed their native context window, resulting in near-zero performances. However, we believe that it is still important to stress test models at long contexts and if they will degenerate near their context window.

For all models, we follow their instructions for long-context extension; for example, we apply the YaRN extension settings for the Qwen3 models as specified on their model card.

For Figure 5, we show the results in terms of z-scores. Every dataset has different means and standard deviation, and the scores span different ranges (e.g., forecasting is almost always between 70 and 80 on our metric, whereas RPS can range from 0 to 100). Consequently, simply taking an unweighted average means the high-variance dataset would dominate the comparison between models. Thus, we first standardize every score to their z-scores first before averaging them for the per-capability metric. Specifically, we take the scores from 8K to 128K across different models for each dataset, and calculate its mean and standard deviation, which are shown in Table 5. Then, we derive the z-score $z = \frac{x-\mu}{\sigma}$.

A.3 Dataset variants

For our main experiments, we decided to stick with one headline setting, but our design enables much deeper and interesting analysis for understanding how models behave in these tasks. For example, we can experiment with different classes of operations as well as present information in different ways.

[13] https://commoncrawl.org/blog/news-dataset-available

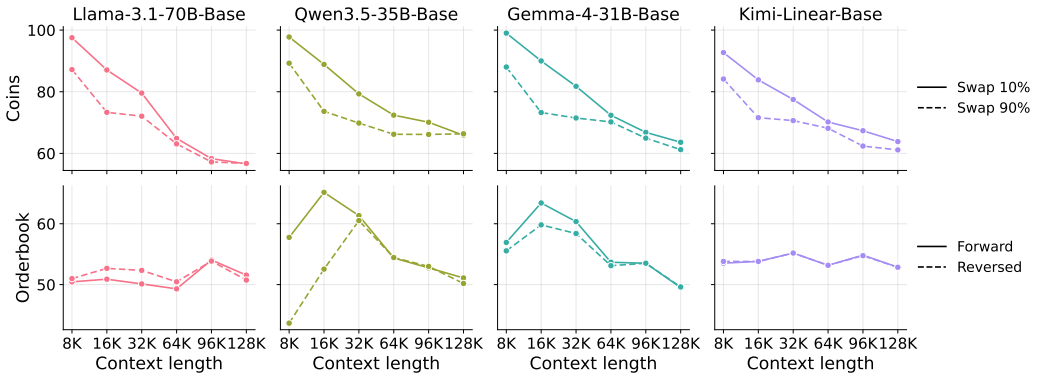

Figure 8: Difference in performance on variants of coins and orderbook. For coins, we increase the probability of swapping as opposed to adding/removing coins from 10% to 90%. For orderbook, we use reverse chronological order.

We ablate design decisions in the Coins and Orderbook datasets for this analysis. For Coins, we increase the probability of swapping any two piles from 10% to 90%, which means that most of the operations go from linear aggregation (i.e., adding/removing a coin from one pile) to non-linear operations (i.e., swapping piles). For Orderbook, we present the order in reverse chronological order instead of forward order, which enables us to analyze if the order of state operations affect the model performance.

We present the results for selected models in Figure 8. We observe that there tends to be a significant gap between the two variants of Coins at shorter input lengths—increasing the percentage of swap operations led to as much as a 50% decrease in performance across the four models. As the context length approaches 128K tokens, both variants converge to roughly the same performance around 60%, which is worse than the naive baseline performance. This suggests that models are better at handling linear aggregation than other affine transformations in state tracking.

Furthermore, we see near-identical performance between the forward and reversed variants of Orderbook, with the exception of Qwen3.5-35B-Base at shorter contexts. The gap between the two variants is mostly under 3 absolute points across models, suggesting that models do not necessarily process state tracking tasks sequentially. Even Kimi-Linear, which leverages recurrent state modules, performs the same on the two variants (Team et al., 2025).

A.4 Results on other benchmarks

We evaluate subsets of HELMET and RULER, which consist of JSON KV from HELMET (Yen et al., 2025), NIAH MK 2, NIAH MK 3, and NIAH MV from RULER (Hsieh et al., 2024). We show the average across these. Additionally, we also evaluate other benchmarks, such as LongBench v2 (Bai et al., 2025), MRCR (Vodrahalli et al., 2024), and GraphWalks (OpenAI, 2025). For MRCR, we evaluate the version with 4 needles released by OpenAI[14]. For GraphWalks, we evaluate the version released by the HELMET authors[15] as the input lengths are measured by Llama 3 tokenizer, and the maximum depth of the BFS variant is 5, and average the performance on the parent and the BFS variants. For LongBench v2, we evaluate the long subset, but truncate all documents to 512K Llama 3 tokens so that the API models do not refuse due to exceeding the model context window.

The full results for the API models at 512K input length are shown in Figure 9. The full results for all models from input lengths 8K to 128K are shown in Table 4. LongBench v2 is excluded from this table because it does not support fine granularity of input lengths.

We evaluate the open-weight models on the same datasets as well, from input lengths 8K to 128K, and the results are shown in Table 4. LongBench v2 is excluded because it does not have the different

[14] https://huggingface.co/datasets/openai/mrcr
[15] https://github.com/princeton-nlp/HELMET/tree/v2

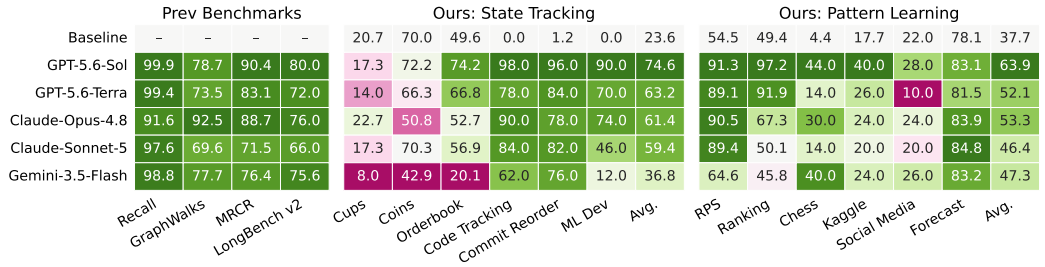

Figure 9: Results of API models at 512K input length, compared with previous synthetic benchmarks, recall (subsets of RULER and HELMET Recall), GraphWalks, MRCR, and LongBench v2. Recall is mostly saturated. Despite the variation in performance, GraphWalks, MRCR, and LongBench v2 similarly only test for recall capabilities by design. LEAPS is conceptually distinct from previous benchmarks.

	Recall					GraphWalks					MRCR					Avg.				
Model	8K	16K	32K	64K	128K	8K	16K	32K	64K	128K	8K	16K	32K	64K	128K	8K	16K	32K	64K	128K
Llama-3.1-8B-Base	99.7	99.6	98.3	88.7	54.0	6.4	2.3	1.6	1.1	2.5	0.0	0.0	0.0	0.0	0.0	58.8	57.6	56.6	51.0	31.6
Llama-3.1-8B-Inst	99.5	98.8	97.5	97.3	72.5	2.5	0.5	8.1	12.8	4.7	38.3	24.5	21.1	32.0	24.6	63.0	60.1	61.0	63.8	46.3
Llama-3.1-70B-Base	**100.0**	99.8	99.0	95.7	49.1	30.1	21.3	12.2	5.4	2.5	0.0	0.0	0.0	0.0	0.0	65.8	63.1	60.1	56.2	28.8
Llama-3.1-70B-Inst	99.9	**99.9**	99.1	97.9	37.8	24.2	47.9	42.3	33.6	7.5	57.2	33.9	28.1	27.9	12.0	72.2	75.6	72.7	69.6	25.5
ProLong-8B-Base	96.8	98.5	97.7	96.6	95.1	10.5	7.7	4.4	0.5	0.1	0.0	0.0	0.0	0.0	0.0	58.3	58.5	57.1	55.3	54.4
ProLong-8B-Inst	95.2	97.0	96.1	97.4	92.0	29.3	23.4	17.1	9.2	12.1	28.7	22.8	17.9	34.4	21.1	66.9	65.4	62.4	63.2	59.1
GPT-OSS-120B	88.2	83.1	86.5	83.5	52.6	75.8	67.1	61.2	44.8	24.1	75.1	54.4	43.1	31.7	19.9	82.8	74.4	73.1	65.0	39.8
Qwen3-8B-Base	84.7	80.5	48.7	39.1	0.0	11.4	4.6	1.7	0.2	29.0	0.0	0.0	0.0	0.0	0.0	51.7	47.3	28.3	22.4	8.3
Qwen3-8B	96.7	90.7	75.1	59.6	0.0	38.3	35.0	13.5	20.0	29.0	0.0	0.0	0.0	0.0	0.0	66.2	61.8	46.8	39.8	8.3
Qwen3-30B-Base	94.5	84.2	64.2	47.1	0.0	8.2	3.0	3.4	2.6	29.0	0.0	0.0	0.0	0.0	0.0	56.3	49.0	37.6	27.6	8.3
Qwen3-30B	98.3	90.6	81.9	63.1	0.0	59.1	30.1	29.3	16.5	29.0	0.0	0.0	0.0	0.0	0.0	73.1	60.4	55.2	40.7	8.3
Qwen3-30B-Think-2507	**100.0**	99.8	99.9	97.8	98.3	74.7	50.1	43.9	32.4	26.1	0.0	0.0	0.0	0.0	0.0	78.5	71.3	69.6	65.1	63.6
Qwen3-30B-Inst-2507	99.8	99.7	99.6	98.2	98.5	33.4	27.6	23.4	21.9	14.0	30.4	24.5	25.4	35.0	26.7	70.9	68.3	67.2	67.4	64.1
Qwen3.5-9B-Base	98.1	98.0	97.0	97.6	96.4	28.0	15.4	12.7	13.8	8.6	0.0	0.0	0.0	0.0	0.0	64.1	60.4	59.0	59.7	57.6
Qwen3.5-9B	95.5	94.8	95.6	96.1	92.2	77.7	74.4	75.7	75.1	73.0	0.0	0.0	0.0	0.0	0.0	76.8	75.4	76.3	76.4	73.6
Qwen3.5-35B-Base	95.9	96.7	97.9	98.0	98.1	28.8	23.6	29.8	29.5	27.9	0.0	0.0	0.0	0.0	0.0	63.0	62.0	64.5	64.4	64.0
Qwen3.5-35B	98.2	97.4	97.2	95.3	93.8	**78.3**	74.7	74.9	**75.9**	**78.3**	0.0	0.0	0.0	0.0	0.0	78.5	77.0	76.9	76.1	75.9
Gemma-4-12B-Base	98.9	99.1	98.6	98.4	95.4	14.9	14.5	11.6	2.5	1.0	0.0	0.0	0.0	0.0	0.0	60.8	60.7	59.6	56.9	54.8
Gemma-4-12B-Inst	99.7	99.6	99.4	98.8	94.5	42.9	43.5	53.8	45.2	33.2	73.3	53.0	48.6	38.9	29.9	79.7	76.9	79.1	74.9	67.7
Gemma-4-31B-Base	99.8	99.8	99.7	99.4	98.8	37.7	34.5	28.5	21.8	9.7	0.0	0.0	0.0	0.0	0.0	67.8	66.9	65.1	63.1	59.3
Gemma-4-31B-Inst	**100.0**	**99.9**	**99.9**	**99.6**	**99.1**	75.1	**77.1**	**76.7**	70.0	42.7	91.1	73.8	71.7	64.7	62.8	**91.6**	**89.7**	**89.3**	**86.1**	77.8
Nemotron-3-30B-Base	97.1	92.9	91.9	81.7	77.0	6.5	2.2	1.2	0.3	0.4	0.0	0.0	0.0	0.0	0.0	57.4	53.7	52.9	46.8	44.1
Nemotron-3-30B	88.8	83.6	67.6	65.8	52.0	49.4	30.5	28.7	26.5	27.8	0.0	0.0	0.0	0.0	0.0	64.8	56.5	46.8	45.2	37.7
Kimi-Linear-Base	98.6	99.1	98.9	99.2	97.7	16.9	11.4	8.5	4.9	4.3	0.0	0.0	0.0	0.0	0.0	61.2	59.9	59.0	58.1	57.1
Kimi-Linear-Inst	99.2	98.8	98.9	98.9	98.4	68.5	66.3	61.2	58.7	55.1	**94.3**	**98.7**	**91.8**	**87.6**	**81.5**	89.8	89.5	87.1	85.8	**83.6**

Table 4: Results of all models on previous benchmarks: Recall is a mix of RULER and HELMET Recall, GraphWalks, MRCR, and the average across these categories. LongBench v2 is excluded as it does not support fine granularity of input lengths.

length buckets that we evaluated on. We do not evaluate at 96K input length as many of the datasets did not generate data at that length.

A.5 ADDITIONAL RESULTS

In this section, we show additional results: the results in z-scores in Figure 10, the full results in Figure 11, the absolute performance on each capability in Figure 12, results of thinking models if they were to finish Figure 13. The t-stats for the decrease in performance from 8K to 128K are 3.9, 10.0, and 7.3 for recall, state tracking, and pattern learning, respectively.

	State Tracking						Pattern Learning					
	8K	16K	32K	64K	96K	128K	8K	16K	32K	64K	96K	128K
Baseline	-0.16	-0.24	-0.32	-0.36	-0.39	-0.40	0.13	0.10	0.00	-0.02	-0.03	-0.04
GPT-5.6-Sol	2.07	2.03	1.45	1.34	1.26	1.28	1.59	1.44	1.52	1.38	1.29	1.24
GPT-5.6-Terra	1.97	1.81	1.32	1.18	1.05	1.09	0.68	0.92	1.00	1.00	0.81	0.93
Claude-Opus-4.8	2.04	1.92	1.70	1.30	1.03	0.82	1.00	0.98	0.88	0.91	1.02	0.91
Claude-Sonnet-5	1.92	1.51	1.60	1.45	0.96	1.00	1.05	0.84	0.71	0.93	0.75	0.71
Gemini-3.5-Flash	2.04	1.56	1.18	1.00	0.62	0.52	1.51	1.36	1.41	1.60	1.48	1.17
Llama-3.1-8B-Base	0.08	-0.31	-0.35	-0.39	-0.43	-0.45	0.17	-0.03	-0.18	-0.11	-0.18	-0.26
Llama-3.1-8B-Inst	-0.20	-0.26	-0.33	-0.34	-0.41	-0.51	-0.29	-0.42	-0.65	-0.59	-0.90	-1.16
Llama-3.1-70B-Base	0.45	0.07	-0.08	-0.27	-0.33	-0.46	0.52	0.54	0.29	0.28	0.13	-0.12
Llama-3.1-70B-Inst	0.51	0.14	-0.11	-0.36	-0.37	-0.45	0.49	0.29	0.25	-0.11	-0.34	-0.73
ProLong-8B-Base	0.15	-0.25	-0.31	-0.38	-0.42	-0.43	0.08	-0.20	-0.13	-0.01	-0.05	-0.08
ProLong-8B-Inst	0.27	-0.16	-0.25	-0.33	-0.38	-0.40	0.28	0.11	0.03	-0.03	-0.10	0.01
GPT-OSS-120B	1.34	0.63	0.30	0.12	-0.16	-0.59	0.34	0.36	0.11	0.23	0.10	-0.68
Qwen3-8B-Base	0.08	-0.19	-0.35	-0.39	-0.46	-1.26	0.40	-0.07	-0.41	-0.39	-0.65	-2.03
Qwen3-8B	-0.05	-0.81	-1.04	-1.12	-1.14	-1.24	-0.45	-0.59	-0.60	-0.66	-1.20	-2.31
Qwen3-30B-Base	0.30	-0.07	-0.23	-0.37	-0.46	-1.25	0.39	0.13	-0.22	-0.30	-0.88	-2.08
Qwen3-30B	0.47	-0.78	-0.99	-1.08	-1.07	-1.22	-0.20	-0.78	-0.81	-0.67	-0.83	-2.29
Qwen3-30B-Think-2507	0.86	0.17	-0.36	-0.65	-0.73	-0.89	-0.30	-0.07	0.09	-0.20	-0.47	-0.27
Qwen3-30B-Inst-2507	0.29	-0.06	-0.14	-0.32	-0.32	-0.40	0.34	0.38	0.04	-0.02	-0.13	-0.12
Qwen3.5-9B-Base	0.60	0.22	0.01	-0.23	-0.31	-0.38	0.36	0.09	0.19	0.06	-0.06	0.04
Qwen3.5-9B	0.34	-0.65	-0.72	-0.86	-0.81	-0.98	-1.39	-1.50	-1.65	-1.90	-2.10	-2.20
Qwen3.5-35B-Base	0.46	0.29	0.09	-0.06	-0.20	-0.20	0.44	0.19	0.13	0.19	0.07	-0.01
Qwen3.5-35B	0.97	-0.23	-0.48	-0.62	-0.67	-0.93	-0.44	-0.58	-0.67	-1.01	-1.23	-1.26
Gemma-4-12B-Base	0.28	0.05	-0.09	-0.26	-0.38	-0.41	0.37	0.28	-0.03	-0.05	0.03	-0.05
Gemma-4-12B-Inst	-0.12	-0.35	-0.40	-0.50	-0.64	-0.62	0.08	0.08	0.06	0.39	0.36	0.32
Gemma-4-31B-Base	0.65	0.37	0.19	0.02	-0.12	-0.20	0.83	0.62	0.52	0.43	0.40	0.44
Gemma-4-31B-Inst	0.31	-0.19	-0.24	-0.35	-0.52	-0.46	0.42	0.17	0.38	0.55	0.46	0.40
Nemotron-3-30B-Base	0.11	-0.22	-0.30	-0.41	-0.50	-0.48	0.16	-0.17	-0.25	-0.16	0.01	-0.15
Nemotron-3-30B	0.05	-0.78	-0.91	-0.99	-1.02	-1.06	-0.55	-0.75	-0.74	-0.50	-0.28	-0.32
Kimi-Linear-Base	0.20	-0.12	-0.21	-0.31	-0.35	-0.41	0.05	-0.04	-0.17	-0.43	-0.31	-0.37
Kimi-Linear-Inst	-0.15	-0.06	-0.13	-0.27	-0.35	-0.36	0.41	0.12	0.00	-0.22	-0.24	-0.31

Figure 10: Results in z-scores averaged across each category from 8K to 128K tokens. We first calculate the mean and variance for a dataset across 30 evaluated models and 6 lengths and normalize all scores into the z-score to reduce bias for any particular dataset in the category average. Absolute scores and all models are shown in Figure 11.

dataset	mean	std
Chess	0.118778	0.122106
Code Tracking	0.297739	0.324307
Coins	0.623230	0.301486
Coins Swap	0.543219	0.283638
Commit Reorder	0.293655	0.334649
Cups	0.257087	0.227134
Forecast	0.710904	0.160736
Kaggle	0.228704	0.088361
ML Dev	0.358389	0.344583
Orderbook	0.428420	0.266986
Orderbook Reversed	0.366124	0.236013
RPS	0.478728	0.283423
Ranking	0.371046	0.321393
Social Media	0.229648	0.055132

Table 5: Mean and standard deviations across all models on lengths from 8K to 128K input lengths.

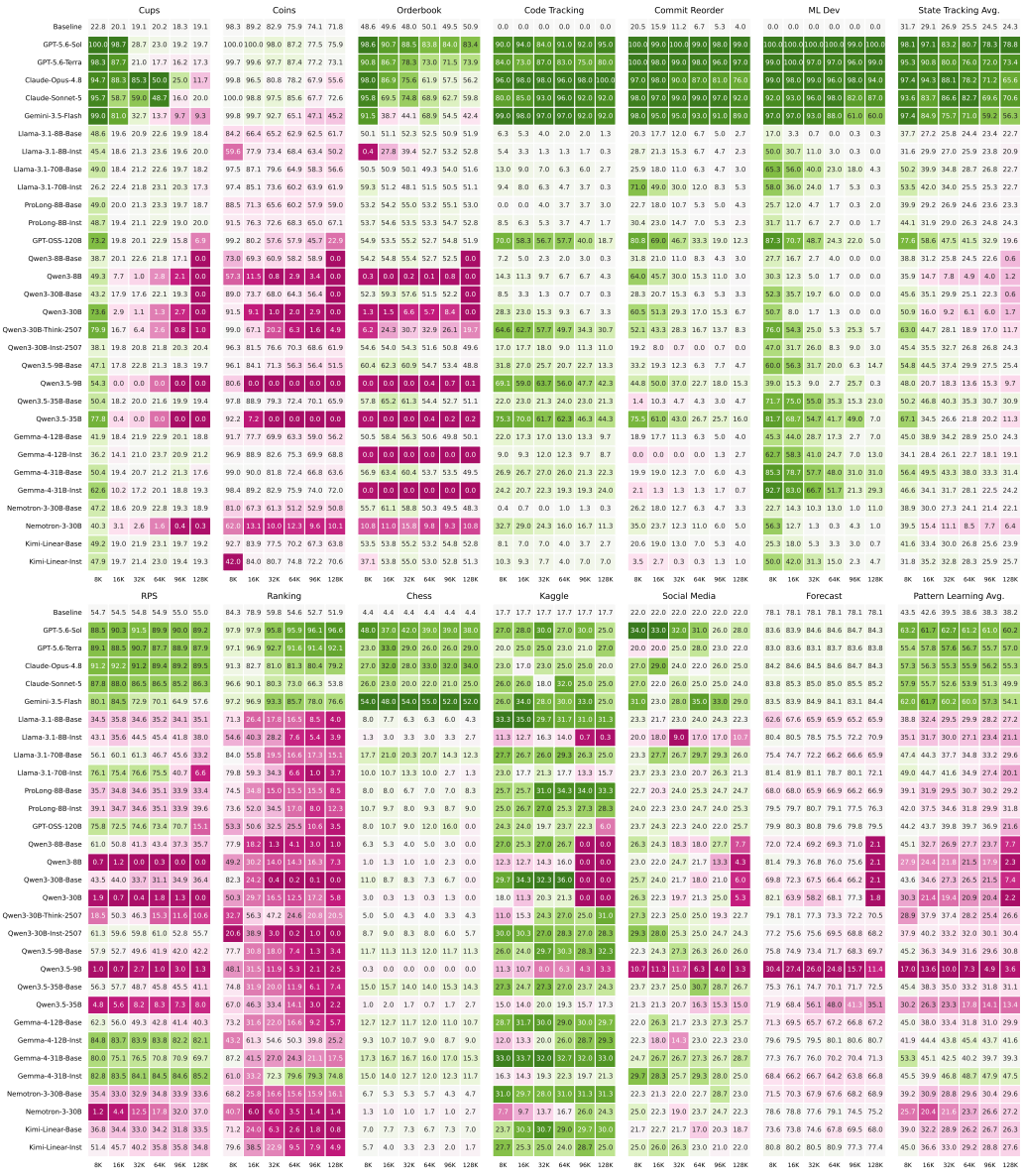

Figure 11: Main results across six datasets for each category and input lengths from 8K to 128K tokens for all 30 evaluated models.

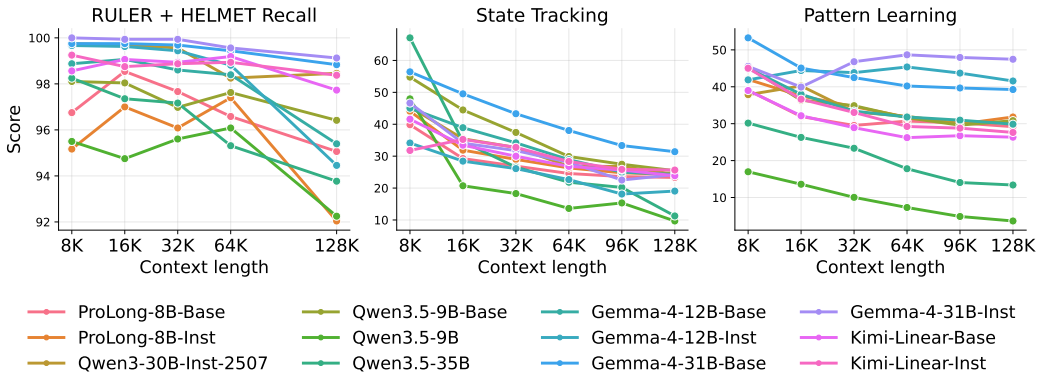

Figure 12: Average absolute performance on each capability. Recall tasks are saturated, and models observe relatively limited degradation, whereas LEAPS tests for new capabilities that are still not saturated and experience more degradation.

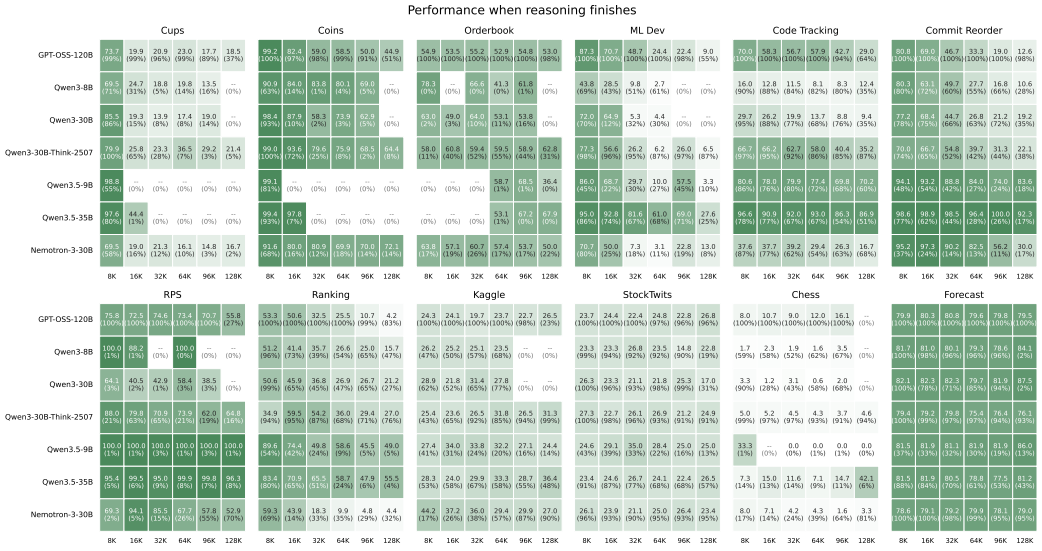

Figure 13: Results of thinking models when thinking finishes. The default generation budget is up to 16K tokens.