TRAINED AGENTIC CONTEXT MANAGEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

We study long context language models. Instead of training long context natively, or designing a long context harness, we train a model over the simplest possible harness: a tool to call itself with any specified prompt and a tool to read tokens in a range from the input context. We finetune Qwen3.6-35B-A3B on a diverse synthetic dataset using this harness. With only 8,000 tokens of context, our small model is as strong as GPT-5.4 with 1M tokens of context on the OOLONG-synth benchmark when document length exceeds 40K tokens.

1 INTRODUCTION

As language models boast impressive breakthroughs (Luong & Lockhart, 2025; Wei, 2025; OpenAI, 2025b; Lin & Cheng, 2025; OpenAI, 2026g;f;e), increasing focus is placed on their limitations. The most salient resource constraint of a language model is its context length. Models can process and produce tokens within this limit, but the limit is hard. Model producers have identified this constraint and placed great emphasis on extending it as large as possible, with some model builders touting 10M (Dangel, 2026; Meta AI, 2025; Zhu et al., 2026) and even 100M (Magic, 2024) token context lengths (for reference, the length of the first Harry Potter book is about 100,000 tokens, the full U.S. tax code roughly 3M, and the Linux kernel source code about 300M).

While native context scaling is appealing as a concrete engineering problem, it is not the only approach to producing a model capable of processing long documents. In 2021, OpenAI fine-tuned GPT-3 to summarize sections of books, and then summarize those summaries, achieving competitive performance with human book summaries on a model with a context length of just 2,048 tokens (Wu et al., 2021). As the field has matured, agentic harnesses have become viable. MemGPT treats context similarly to an operating system, relying on LLM-invoked function calls to push relevant information in and out of the context window to external memory, effectively achieving expanded virtual context (Packer et al., 2023). More recently, Recursive Language Models introduced a harness that utilizes LLM's coding prior to call itself in a REPL environment, achieving stronger performance than base models on retrieval and aggregation tasks both within context limits, but also beyond it, expanding effective context by $\sim$4x (Zhang et al., 2025).

While handcrafted prompts and programmatic environments can be effective in solving long context problems without native long context training, it begs an obvious question: if models can, for example, solve all the problems at the International Collegiate Programming Contest[1] World Finals (OpenAI, 2025b), or more generally, implement requested features in codebases far larger than context limits (Marten et al., 2026; OpenAI, 2026d; Anthropic, 2026b), can they not be *trained* to manage context effectively, *without human-authored harness crutches*? Such model training internalizes decisions external harnesses hardcode, offering the potential to scale much more generally.

We answer this question in this paper, affirmatively. We consider the simplest possible agentic context harness, defined below. The purpose of the harness is to *enable capability*, without *fixing choices* or *instilling a prior*.

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.

[1] https://icpc.global/

1.1 HARNESS

Our harness has two tools:

1. `read_chunk(start, end)`: Read a slice of the document and return the decoded text of tokens [`start`, `end`).
2. `spawn_subagent(subtask)`: Spawn a fresh-context copy of yourself to solve `subtask`; returns the child's `\boxed{}` answer.

Unmodified long-context prompts are provided to the model over a document accessed via the above harness. The system prompt is propagated to child agents and includes the above harness definition, a generic instruction, and potentially task-specific background, in accordance with the respective eval[2]. Subagents' user prompt is specified by the calling model.

1.2 RESULTS

We finetune Qwen3.6-35B-A3B on a diverse synthetic dataset using this harness and achieve the following results:

1. On the OOLONG-synth benchmark, with a context limit of just 8K tokens per agent, our small model performs **as well** as GPT-5.4 with 1M tokens of context when document length exceeds 80K tokens.
2. On the RULER benchmark, with a context limit of just 8K tokens per agent, our model scores $\geq 85\%$ on document size from 10K to 320K tokens.

1.3 PAPER ORGANIZATION

In the next section, we survey related work to frame the contribution of the following sections. In Section 3, we describe our training methodology. In Section 4, we discuss our experiments, and we conclude in Section 5.

2 RELATED WORK

Long-context LLM work can be broken into a few distinct categories: *native long context models*, *context compaction*, and *long context harnesses*, both trained and untrained.

2.1 NATIVE LONG CONTEXT MODELS

GPT-2 shipped with a 1,024 token context limit (Radford et al., 2019). This limit doubled in every generation through GPT-4 (Brown et al., 2020; OpenAI, 2022; 2023b), until jumping to the 100K range on GPT-4 Turbo (OpenAI, 2023a), with contemporaneous models Claude 2 (Anthropic, 2023) and later Llama 3.1 (Grattafiori et al., 2024) at similar limits. The core difficulty in scaling context length n in the classic Transformer architecture (Vaswani et al., 2017) is $\Omega(n^2)$ attention complexity, which becomes prohibitively expensive as $n >> 100K$. State-space architectures like the mamba lineage (Gu & Dao, 2024; Dao & Gu, 2024; Lahoti et al., 2026), neural methods like Titans (Behrouz et al., 2024), and Linear Transformer (Katharopoulos et al., 2020) derivatives DeltaNet (Schlag et al., 2021) and Gated DeltaNet (Yang et al., 2025) abandon quadratic attention for recurrence. Modern architectures are hybrid, combining linear or sparse attention layers with dense attention, allowing context lengths to scale to 1M+ tokens while retaining long-context performance (Qwen Team, 2026; Kimi Team, 2026b; DeepSeek-AI, 2025; GLM-5 Team, 2026; NVIDIA, 2025; Gemma Team, 2024; Mistral AI, 2024).

2.2 CONTEXT COMPACTION

An alternative to native long context, and an inevitability when such native context fills, is a function f mapping current context x to a reduced context y where $|y| << |x|$. Three categories of

[2]Precise harness environment is given in Appendix A.

such lossy f exist: *masking*, *summarization*, and *learned compaction*. The simplest compaction technique is masking: dropping tokens based on a predetermined rule. Clearing tool call results is a popular recommended choice (Anthropic, 2025a), achieving results competitive with summarization (DeepSeek-AI, 2025; Lindenbauer et al., 2025). Alternative approaches evict tokens based on attention statistics (Xiao et al., 2024; Zhang et al., 2023; Li et al., 2024). Summarization is the current production harness default, implemented in Claude Code, Codex, and Gemini CLI (Anthropic, 2025a; OpenAI, 2026a; Google, 2026; Barbaste et al., 2026). Trained compaction is a new frontier, described as part of GPT-5.1-Codex-Max (OpenAI, 2025a) and Cursor's composer (Cassano & Rush, 2026) training. "Still" trains a model to compress KV-Cache directly (O'Neill et al., 2026).

2.3 Long Context Harnesses

As long as LLMs have proven useful, an evolving line of work has focused on how to expand their capabilities without touching model internals. The first efforts were known as prompt engineering (Phoenix & Taylor, 2024; Berryman & Ziegler, 2024), which, though necessary on pre-trained base models (Brown et al., 2020), became internalized through instruction-tuning and reasoning RL (Ouyang et al., 2022; Guo et al., 2025). Function-calling has lead to a new wave of such efforts, where domain-specific knowledge, tools, and multi-agent systems live on a scaffold that interacts with powerful LLMs known as a "harness" (Lopopolo, 2026; Barbaste et al., 2026).

Long-context harnesses originate with systems like MemGPT (Packer et al., 2023), RAPTOR (Sarthi et al., 2024), and Chain of Agents (Zhang et al., 2024), where the overall goal is to engineer workflows that decompose documents into contexts independent LLM calls can read and aggregate. Recursive Language Models is a popular recent example that allows for single-level recursive LLM calls via a programmatic environment (Zhang et al., 2025). Follow-up work expands subagent capabilities (Lumer et al., 2026), trains shared parent/child policies (Kim & Ahmad, 2026), and incorporates self-reflection (Alizadeh et al., 2026).

Memagent (Yu et al., 2026) trains a harness that walks linearly through context, using RL and a harness prompt to determine what to carry from node-to-node. Similarly to our work, they also use a small (8K token) context while processing long (as much as 3.5M token) documents. However, they test primarily on the distribution they train, compare only against other small models, and hardcode execution path and prompts in the harness.

Frontier models now utilize multi-agent systems. Kimi K2.5 trains "agent swarm" (Kimi Team, 2026a): the model is trained to call subagents via reinforcement learning. Subagent trajectories are not trained. GPT-5.6 has also trained multi-agent behavior (OpenAI, 2026c), notably utilizing 10,000 agents to solve Navier-Stokes (OpenAI, 2026e). Anthropic, DeepMind, Meta, and xAI likely do similarly, with multi-agent systems touted in respective model and product releases (Anthropic, 2025b; 2026a; OpenAI, 2026b; xAI, 2026; 2025; Meta Superintelligence Labs, 2026; Google, 2025; Gottweis et al., 2026).

3 Methods

We deliberately use the minimal tool definitions necessary to train agentic context management. However, our harness is sufficient to allow arbitrarily complex multi-agent decompositions. Decompositions in the literature fall into roughly three categories: *tree reduce* (Wu et al., 2021; Sarthi et al., 2024; Zhang et al., 2022), *sequential folds* (Yu et al., 2026; Zhang et al., 2024), and *agentic delegation* (Zhang et al., 2025; Kimi Team, 2026a; Anthropic, 2025b). We utilize all three strategies.

Without any training, models generally make poor use of our harness. Base Qwen3.6-35B-A3B, GPT-5.4, and Kimi K2.6 use subagent delegation sporadically, sometimes choosing to read the entire document in one context, even if context limit is specified to be less than document length, and other times delegating the exact original question to subagents. Opus 4.6 does slightly better, showing real context pressure awareness and document range subagent delegation. Unfortunately it delegates *too* much, blowing through $300 of API credits building massive subagent trees on a 100-question eval,

only to overcount[3]. Instead, we compare to base models supplied the entire document in context with native context limits, where performance is stronger.

In general, models have a *weak prior* for context decomposition. Pre-training is pure language continuation, and agentic post-training does not teach decomposition strategies. Where Recursive Language Models (Zhang et al., 2025) can rely on strong coding priors instilled in both pre- and post-training, decomposition on our harness needs to be taught.

One idea might be to RL an open weight model on our harness. Unfortunately, exploration in LLM RL is far too weak to escape the weak decomposition prior at modest compute budgets. It is also unclear what reward to assign subagents, and if GRPO (Shao et al., 2024) or its derivatives (Yu et al., 2025; Liu et al., 2025; Zheng et al., 2025) are used, what group to compare subagent trajectories in.

We instead SFT a decomposition strategy. We train *tree reduce* for order-independent tasks (see Figure 1), *sequential fold* for order-dependent tasks (see Figure 2), and *agentic delegation* to handle boundary conditions whenever we read the document (see Figure 3). By not hardcoding a decomposition structure in our harness, our model internalizes and composes delegation strategies according to the task and current problem-solving state.

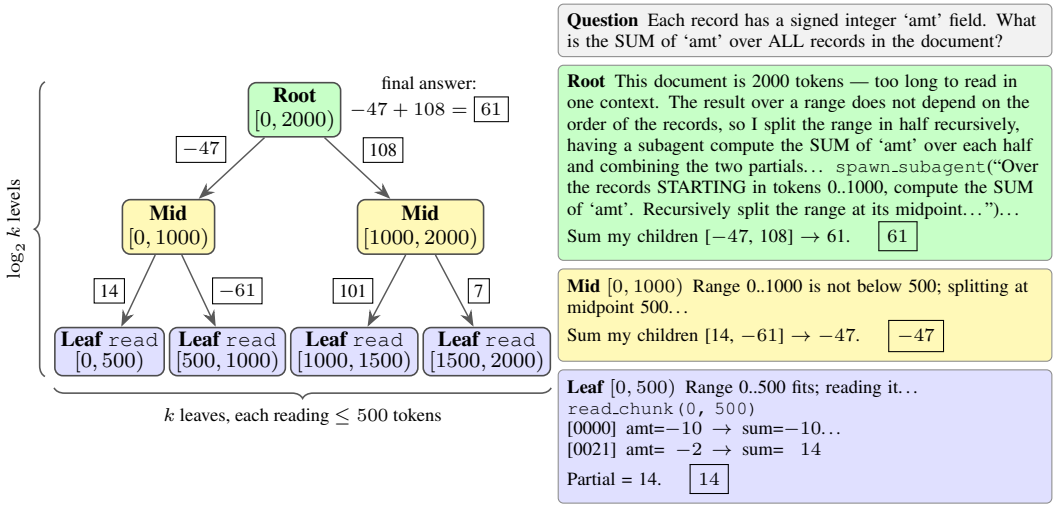
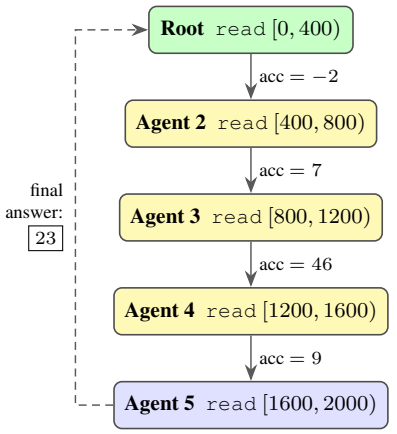


Figure 1: **Tree reduce**, for order-independent tasks (aggregation, retrieval). Non-leaf nodes delegate half their range to subagents. Leaves read one chunk and compute a leaf op which is aggregated up the tree.

[3]Opus 4.6 does appear to perform somewhat decently on our harness with unlimited context budget. Unfortunately, due to the massive subagent trees, running the full comparison without restriction is cost-prohibitive.



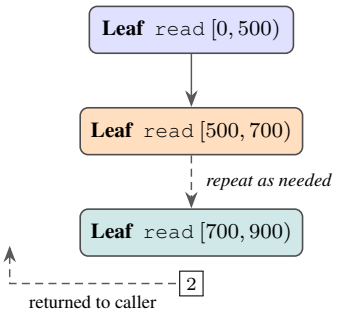
Question Process the records in order, keeping a running total of 'amt'. Whenever a record has grp=RST, reset the running total to 0 (that record's amt is NOT added). What is the final running total after the last record?...

Root This document is 2000 tokens — too long to read in one context. This task is order-dependent: a grp=RST record resets the total, and what a record contributes depends on the resets before it — so I process the document from the start with a running accumulator... `read_chunk(0, 400)`
[0000] amt=−10 → total=−10...
[0005] grp=RST → RESET, total= 0...
[0017] amt= +14 → total= −2
→ accumulator = −2. Delegating the rest 400..2000 with the updated accumulator. `spawn_subagent`("Continue the running accumulator... over the records STARTING in tokens 400..2000. accumulator so far = −2...")

Agent 2 Continue the running accumulator...over the records STARTING in tokens 400..2000. accumulator so far = −2... `read_chunk(400, 800)`
[0018] amt= −2 → total=−4...
[0020] grp=RST → RESET, total= 0...
[0034] amt= +9 → total= 7
→ accumulator = 7. Delegating the rest 800..2000 with the updated accumulator.

Agent 5 ...accumulator so far = 9...Range 1600..2000 fits and reaches the document end; reading it. `read_chunk(1600, 2000)`
[0070] amt= +18 → total=27...
[0086] amt=−17 → total=23
→ accumulator = 23 (final). 23

Figure 2: **Sequential fold**, for order-dependent tasks (variable tracking, running state). Each agent reads the next slice, updates the accumulator, and delegates the rest of the document to the next agent.



Question How many entries contain the word 'good' anywhere in their body text?

Leaf [0, 500) Range 0..500 fits; reading it, then the next 200 tokens to finish the last entry's body (it may run well past 500).
`read_chunk(0, 500)` → *Entry: 1... Entry: 2... the pity of the* ***good*** *God?"... Entry: 3... his are not*
`read_chunk(500, 700)` → *amenable to mere strength.... "Now," he said, "friend*

The last entry's body is still cut off at 700 (no blank line / next header yet), so I read the next 200 tokens (700..900) to finish it.
`read_chunk(700, 900)` → *John, where are the skeleton keys?"... standing him in* ***good*** *stead.... it slowly opened. Entry: 4...*

Computing the partial over the entries whose HEADER STARTS in 0..500 (...any entry whose header starts at/after 500 belongs to the next range):
- entry 1: 'good' appears 0x in the body (no) → count=0
- entry 2: 'good' appears 1x in the body (counts) → count=1
- entry 3: 'good' appears 1x in the body (counts) → count=2

Partial = 2. 2

Figure 3: **Boundary handling.** A leaf/fold agent owns the data that *starts* in its range. It reads 200 additional tokens at a time until the last item in its range is not cut off (additional calls not pictured in Figures 1 and 2, for brevity).

We generate synthetic oracle demonstration data on a wide distribution of aggregation, retrieval, and sequential tasks. While we cover similar shapes to our evals, our training distribution uses independent questions and vastly more diverse examples. The technical challenge is to instill the necessary prior so the model knows how to take a question and delegate appropriate subtasks to agents from which the final answer can be determined.

For example, consider the training question "In how many sections are there STRICTLY more named than unnamed sentences?" (the prompt defines a *named* sentence as a sentence with a proper noun). To answer this question, each leaf must compute a tally of named and unnamed sentences per section in its assigned range. Thus, the root must recognize the necessary, often multi-dimensional, key space and dictate this and a recursive strategy to mid-level agents, who pass the instruction down

to other mid-level agents, until leaves compute their range and counts are aggregated back up the tree, culminating in a final judgement at the root.

Our training data is primarily record-based, but also includes real prose, dot-separated word lists, and haystacks with injected needles. We finetune Qwen3.6-35B-A3B, chosen as a capable small model with affordable compute, trained on the Tinker platform (Thinking Machines Lab, 2025a;b). For more details on our complete training data and training hyperparameters, see Appendices B and C, respectively.

We do consider an RL stage following SFT. Unfortunately, even with LLM-as-a-judge rewards on subagent trajectories, as described in Recursive Agent Optimization Gandhi et al. (2026), the policy exhibits high-variance and latches onto simpler strategies that weaken overall performance. We suspect RL to be useful in learning recovery and strengthening generalization; however, to do this properly probably requires group-level baselines on subagent trajectories, to cut variance, and compute beyond the scope of this paper.

4 Experiments

We test our finetune on RULER (Hsieh et al., 2024), the standard long context eval, and OOLONG-synth (Bertsch et al., 2025), a newer long context benchmark frontier models have not yet saturated, also used in Recursive Language Models (Zhang et al., 2025), Recursive Agent Optimization (Gandhi et al., 2026), and Recursive Agent Harnesses (Lumer et al., 2026). RULER is an extension of needle-in-a-haystack style tasks with variable tracking, word frequencies, and Q&A. OOLONG-synth composes classification tasks on ten datasets into long documents on which aggregation is queried.

We compare against base Qwen3.6-35B-A3B[4] and GPT-5.4. We chose GPT-5.4 as a selection from the best frontier models available upon Qwen3.6-35B-A3B release. While we do not eval against additional frontier models, due to compute requirements, we recommend the reader read the OOLONG paper for more comparisons (GPT-5 tops the chart) (Bertsch et al., 2025). While we do eval GPT-5.4 on RULER, this benchmark is largely saturated for frontier models. For more RULER reference points, consult model release cards.

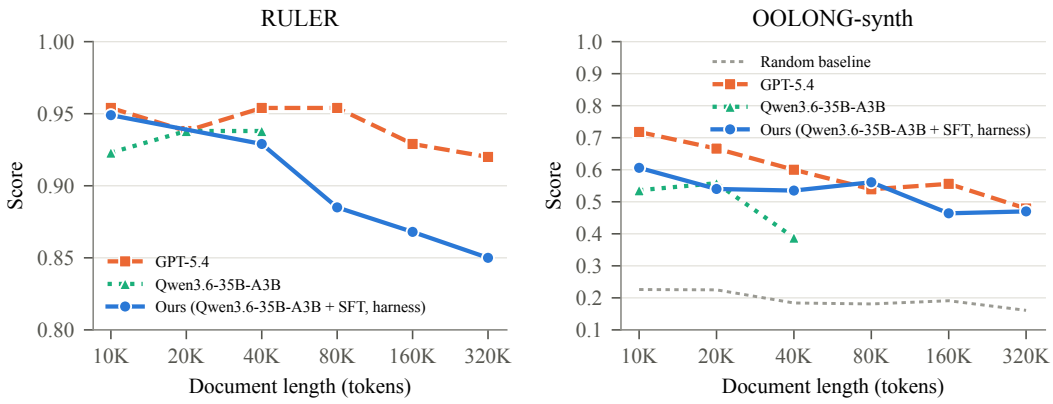


Figure 4: **Results** on RULER and OOLONG-synth. Our finetune runs in our harness with an 8K token per-agent context limit. GPT-5.4 (medium reasoning) and base Qwen3.6-35B-A3B are given the full document with the question (see Appendix A for details). Base Qwen and GPT-5.4 are served with 64K and 1M token context limits, respectively. The OOLONG random baseline follows Bertsch et al. (2025, §2.3), computed on the same problems.

For RULER, we fix a sample of 5 questions per problem type, for a total of 65 problems per document length. For OOLONG-synth, we fix a sample of a 10 questions per family: counting, user, and

[4]While Qwen3.6-35B-A3B was trained with a 256K token context limit, we sample Qwen3.6-35B-A3B from Tinker, which caps the model at 64K token context.

temporal, each sampled from a different underlying classification dataset. We vendor generators for both benchmarks and generate our problems from scratch.

On RULER, our finetune performs worse than GPT-5.4, while remaining above 85%. On OOLONG-synth, our model starts worse than GPT-5.4, then matches its performance when document size exceeds 80K. At this length, native context performance begins to degrade, and a single model can be overwhelmed with the number of items requiring tracking within its context. On the other hand, agents in our harness operate in 8K token budgets, executing small pieces of an overall decomposition strategy, which scales gracefully to arbitrary lengths.

Per problem-family results are given in Table 1.

RULER	10K	40K	80K	160K	320K
NIAH (8 tasks)	1.00	1.00	1.00	1.00	1.00
VT	1.00	1.00	0.92	0.84	0.70†
CWE	1.00	0.74	0.58	0.24	0.02
FWE	0.93	0.93	1.00	1.00	0.93
QA-1	0.60	0.80	0.60	0.60	0.60
QA-2	0.80	0.60	0.40	0.60	0.80
Mean (13 tasks)	0.949	0.929	0.885	0.868	0.850†

OOLONG	10K	20K	40K	80K	160K	320K
Counting	0.45	0.42	0.42	0.41	0.41	0.40
User	0.86	0.72	0.80	0.73	0.61	0.60
Temporal	0.51	0.48	0.39	0.54	0.37	0.41
Mean	0.606	0.540	0.535	0.561	0.464	0.470

Table 1: **Results by task family**. **Left:** On RULER, our model chooses a lossy aggregation strategy for the common words "CWE" problem type, which degrades when common words are diluted by fillers as document length grows. GPT-5.4 scores 0.56 on CWE at doc size 320K. **Right:** On OOLONG-synth, leaf classification and merge errors compound at length. Distribution mismatch also arises at 160K+ document length (it sees $\leq 14K$ doc size in training). †4 of 5 problems.

5 Conclusion

In this paper we show training on a bare bones agentic harness is sufficient for a small model with small context to perform as well as a frontier model with 1M token context on long-context evals. While we do not claim full generalization of our finetune, we also do not train on our evaluation distributions. The point of this paper is to show the power of agentic training in contrast to harness engineering, excessive native long context, and lossy compaction. As a side effect we also show the power of synthetic oracle SFT data generated by powerful frontier models (Opus 4.8, 5, 5.5, and Fable 5 and 5.1, in this case).

The thesis of the last decade of AI development is that specific model capability can be incorporated into fully-general models by training on a wider distribution. The methods of this paper can be incorporated into frontier model development, so such models exhibit improved agentic performance under context pressure, without wasting tokens. This is in contrast to an external harness which must be applied at runtime.

To develop our curriculum into a production recipe, the training distribution would need to be widened and a reinforcement learning stage would likely prove fruitful in supercharging agentic capabilities. We leave this as important future work, outside the scope of this paper or a small research project more generally.

AI use statement

In this work, we used generative AI tools to write all our code, administer experiments, create figures, search and analyze relevant literature, format references, generate synthetic data sets, propose or refine hypotheses/brainstorming, design or provide feedback on research methodology/experiments, clean and reformat data, support qualitative and thematic data analysis, and interpret results. We have not used generative AI tools to draft parts of this research paper, and the following topics are not applicable to this work: help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, and assist with translation. We have reviewed our AI-assisted work, for example, by reviewing AI code and conducting manual audits of synthetic data. We take

responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Keivan Alizadeh, Parshin Shojaee, Minsik Cho, and Mehrdad Farajtabar. Recursive language models meet uncertainty: The surprising effectiveness of self-reflective program search for long context, 2026. URL https://arxiv.org/abs/2603.15653.

Anthropic. Claude 2. Anthropic blog post, July 2023. URL https://www.anthropic.com/news/claude-2.

Anthropic. Effective context engineering for AI agents. Engineering blog post, September 2025a. URL https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents.

Anthropic. How we built our multi-agent research system. Engineering blog post, June 2025b. URL https://www.anthropic.com/engineering/multi-agent-research-system.

Anthropic. Orchestrate teams of Claude Code sessions. Claude Code documentation, 2026a. URL https://code.claude.com/docs/en/agent-teams. Agent teams (experimental), introduced with Claude Opus 4.6, February 2026; accessed 2026-09-23.

Anthropic. Introducing Claude Fable 5.1 and Claude Mythos 5.1. Anthropic blog post, September 2026b. URL https://www.anthropic.com/claude-fable-and-mythos-5-1.

Paul Barbaste, Tristan Darrigol, Germain Vu, and Tom Wiltberger. Harness engineering: Anatomy, architecture, and evolution of coding agents – a source-code study of eleven systems, 2026. URL https://arxiv.org/abs/2609.00006.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time, 2024. URL https://arxiv.org/abs/2501.00663.

John Berryman and Albert Ziegler. *Prompt Engineering for LLMs: The Art and Science of Building Large Language Model–Based Applications*. O'Reilly Media, November 2024. URL https://www.oreilly.com/library/view/-/9781098156145.

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

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. URL https://arxiv.org/abs/2005.14165.

Federico Cassano and Sasha Rush. Training Composer for longer horizons. Cursor blog post, March 2026. URL https://cursor.com/blog/self-summarization.

Justin Dangel. Introducing SubQ: The first fully subquadratic LLM. Subquadratic blog post, May 2026. URL https://subq.ai/introducing-subq. 12M-token research result; the shipped model is SubQ 1M-Preview.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2405.21060.

DeepSeek-AI. DeepSeek-V3.2: Pushing the frontier of open large language models, 2025. URL https://arxiv.org/abs/2512.02556.

Apurva Gandhi, Satyaki Chakraborty, Xiangjun Wang, Aviral Kumar, and Graham Neubig. Recursive agent optimization, 2026. URL https://arxiv.org/abs/2605.06639.

Gemma Team. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.

GLM-5 Team. GLM-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Google. Gemini 3 Deep Think is now available in the Gemini app. Google blog post, December 2025. URL https://blog.google/products/gemini/gemini-3-deep-think/.

Google. Gemini CLI. GitHub repository, Apache-2.0, 2026. URL https://github.com/google-gemini/gemini-cli. Commit f50ba8608e, accessed 2026-09-23. Compression: packages/core/src/context/chatCompressionService.ts.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, et al. Accelerating scientific discovery with Co-Scientist. *Nature*, 655:487–496, 2026. URL https://doi.org/10.1038/s41586-026-10644-y. Preprint: arXiv:2502.18864, "Towards an AI co-scientist".

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The Llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783. Llama 3.1, 128K context.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *Conference on Language Modeling (COLM)*, 2024. URL https://arxiv.org/abs/2312.00752.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638, 2025. URL https://doi.org/10.1038/s41586-025-09422-z. Preprint: arXiv:2501.12948, "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning".

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling (COLM)*, 2024. URL https://arxiv.org/abs/2404.06654.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning (ICML)*, 2020. URL https://arxiv.org/abs/2006.16236.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations (ICLR)*, 2017. URL https://arxiv.org/abs/1609.04836.

Daniel Kim and Rehaan Ahmad. Reinforcing recursive language models. alphaXiv blog post, May 2026. URL https://www.alphaxiv.org/blog/reinforcement-learning-for-rlms.

Kimi Team. Kimi K2.5: Visual agentic intelligence, 2026a. URL https://arxiv.org/abs/2602.02276.

Kimi Team. Kimi K3: Open frontier intelligence. Technical report, Moonshot AI, July 2026b. URL https://github.com/MoonshotAI/Kimi-K3/blob/main/k3_tech_report.pdf. 2.8T-parameter MoE (104B active), 1M-token context; hybrid Kimi Delta Attention and full MLA (69 + 24 layers).

Aakash Lahoti, Kevin Y. Li, Berlin Chen, Caitlin Wang, Aviv Bick, J. Zico Kolter, Tri Dao, and Albert Gu. Mamba-3: Improved sequence modeling using state space principles. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2603.15569.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2404.14469.

Hanzhao Lin and Heng-Tze Cheng. Gemini achieves gold-medal level at the International Collegiate Programming Contest World Finals. Google DeepMind blog post, September 2025. URL https://deepmind.google/blog/gemini-achieves-gold-medal-level-at-the-international-collegiate-programming-conte 10 of 12 problems; would rank 2nd among university teams.

Tobias Lindenbauer, Igor Slinko, Ludwig Felder, Egor Bogomolov, and Yaroslav Zharov. The complexity trap: Simple observation masking is as efficient as LLM summarization for agent context management. In *Fourth Deep Learning for Code Workshop (DL4C) at NeurIPS*, 2025. URL https://arxiv.org/abs/2508.21433.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective, 2025. URL https://arxiv.org/abs/2503.20783. Introduces Dr. GRPO.

Ryan Lopopolo. Harness engineering: Leveraging Codex in an agent-first world. OpenAI blog post, February 2026. URL https://openai.com/index/harness-engineering/.

Elias Lumer, Sahil Sen, Kevin Paul, and Vamse Kumar Subbiah. Recursive agent harnesses, 2026. URL https://arxiv.org/abs/2606.13643.

Thang Luong and Edward Lockhart. Advanced version of Gemini with Deep Think officially achieves gold-medal standard at the International Mathematical Olympiad. Google DeepMind blog post, July 2025. URL https://deepmind.google/blog/advanced-version-of-gemini-with-deep-think-officially-achieves-gold-medal-standard 35/42, five of six problems, graded by IMO coordinators.

Magic. 100M token context windows. Blog post, August 2024. URL https://magic.dev/blog/100m-token-context-windows. LTM-2-mini; introduces the HashHop evaluation.

Martin Marek, Sanae Lotfi, Aditya Somasundaram, Andrew Gordon Wilson, and Micah Goldblum. Small batch size training for language models: When vanilla SGD works, and why gradient accumulation is wasteful, 2025. URL https://arxiv.org/abs/2507.07101.

Ryan Marten, Alex Shaw, Ivan Bercovich, Benedikt Droste, Tommaso Cerruti, Steven Dillmann, Mike Merrill, Andy Konwinski, and Ludwig Schmidt. Terminal-Bench: A benchmark for AI agents in terminal environments. Harbor Framework and Laude Institute, August 2026. URL https://www.tbench.ai/. Version 4.0.0, released 2026-08-26. Zenodo DOI 10.5281/zenodo.22105680.

Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks, 2018. URL https://arxiv.org/abs/1804.07612.

Meta AI. The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation. Blog post, April 2025. URL https://ai.meta.com/blog/llama-4-multimodal-intelligence/. Llama 4 Scout, 10M-token context; model card at https://huggingface.co/meta-llama/Llama-4-Scout-17B-16E-Instruct.

Meta Superintelligence Labs. Introducing Muse Spark: Scaling towards personal superintelligence. Meta AI blog post, April 2026. URL https://ai.meta.com/blog/introducing-muse-spark-msl/. Contemplating mode orchestrates multiple agents that reason in parallel.

Mistral AI. Un Ministral, des Ministraux. Mistral AI blog post, October 2024. URL https://mistral.ai/news/ministraux. Introduces Ministral 3B and Ministral 8B; the latter uses interleaved sliding-window attention.

NVIDIA. NVIDIA Nemotron 3: Efficient and open intelligence, 2025. URL https://arxiv.org/abs/2512.20856.

Charles O'Neill, Alex Sandomirsky, Harry Partridge, Mudith Jayasekara, and Max Kirkby. Still: Amortized KV cache compaction in a single forward pass, 2026. URL https://arxiv.org/abs/2606.07878.

OpenAI. Introducing ChatGPT. OpenAI blog post, November 2022. URL https://openai.com/index/chatgpt/. Launch of the GPT-3.5 series.

OpenAI. New models and developer products announced at DevDay. OpenAI blog post, November 2023a. URL https://openai.com/index/new-models-and-developer-products-announced-at-devday/. Introduces GPT-4 Turbo with a 128K-token context window.

OpenAI. GPT-4 technical report, 2023b. URL https://arxiv.org/abs/2303.08774.

OpenAI. Building more with GPT-5.1-Codex-Max. OpenAI blog post, November 2025a. URL https://openai.com/index/gpt-5-1-codex-max/.

OpenAI. Our general-purpose reasoning models solved all 12 problems at the 2025 ICPC world finals. OpenAI announcement (X), September 17, 2025, September 2025b. URL https://x.com/OpenAI/status/1968368133024231902. 12/12 under the five-hour contest rules; the best human team solved 11.

OpenAI. Codex CLI. GitHub repository, Apache-2.0, 2026a. URL https://github.com/openai/codex. Commit 5f371ba30a, accessed 2026-09-23. Compaction prompt: codex-rs/prompts/templates/compact/prompt.md.

OpenAI. Subagents. Codex documentation, 2026b. URL https://developers.openai.com/codex/subagents. Generally available March 2026; accessed 2026-09-23.

OpenAI. GPT-5.6: Frontier intelligence that scales with your ambition. OpenAI blog post, July 2026c. URL https://openai.com/index/gpt-5-6/. Public release July 9, 2026 (Luna, Terra, Sol); introduces multi-agent mode with concurrent subagents.

OpenAI. Introducing GPT-6 Astra. OpenAI blog post, September 2026d. URL https://openai.com/index/gpt-6-astra/.

OpenAI. Finite time blowup for Navier–Stokes. Manuscript, 166 pages, with Lean formalization; announced September 8, 2026, September 2026e. URL https://cdn.openai.com/pdf/32d9f210-8b73-45e0-91bc-82a30aef8a9a/navier-stokes.pdf. Forced finite-time blow-up with bounded kinetic energy (alternative (C) of the Clay problem statement); announcement at https://openai.com/index/navier-stokes-solution/.

OpenAI. Ten advances in mathematics and theoretical computer science. OpenAI blog post, August 1, 2026, August 2026f. URL https://openai.com/index/ten-advances-in-mathematics/. Astra; ten problems open for ten or more years, each with a Lean 4 proof certificate.

OpenAI. An OpenAI model has disproved a central conjecture in discrete geometry. OpenAI blog post, May 2026g. URL https://openai.com/index/model-disproves-discrete-geometry-conjecture/. Disproof of the Erdős unit-distance conjecture: an infinite family of constructions giving a polynomial improvement over the square-grid bound, verified by external mathematicians.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2203.02155.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems, 2023. URL https://arxiv.org/abs/2310.08560.

James Phoenix and Mike Taylor. *Prompt Engineering for Generative AI: Future-Proof Inputs for Reliable AI Outputs*. O'Reilly Media, May 2024. URL https://www.oreilly.com/library/view/prompt-engineering-for/9781098153427/.

Qwen Team. Qwen3.5: Towards native multimodal agents. Qwen blog post, February 2026. URL https://qwen.ai/blog?id=qwen3.5. First open-weight release Qwen3.5-397B-A17B; hybrid Gated DeltaNet and gated full attention (3:1) with sparse MoE.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2401.18059.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning (ICML)*, 2021. URL https://arxiv.org/abs/2102.11174.

John Schulman and Thinking Machines Lab. LoRA without regret. Thinking Machines Lab blog, 2025. URL https://thinkingmachines.ai/blog/lora/. September 29, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300. Introduces Group Relative Policy Optimization (GRPO).

Thinking Machines Lab. Announcing Tinker. Thinking Machines Lab blog, 2025a. URL https://thinkingmachines.ai/news/announcing-tinker/. October 1, 2025.

Thinking Machines Lab. Tinker Cookbook: Post-training with Tinker. GitHub repository, 2025b. URL https://github.com/thinking-machines-lab/tinker-cookbook.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. URL https://arxiv.org/abs/1706.03762.

Alexander Wei. An experimental reasoning LLM achieves gold medal-level performance on the International Math Olympiad. OpenAI announcement (X thread), July 19, 2025, July 2025. URL https://x.com/alexwei_/status/1946477742855532918. 35/42 under contest conditions; proofs graded by three former IMO medalists.

Jeff Wu, Long Ouyang, Daniel M. Ziegler, Nisan Stiennon, Ryan Lowe, Jan Leike, and Paul Christiano. Recursively summarizing books with human feedback, 2021. URL https://arxiv.org/abs/2109.10862.

xAI. Grok 4. xAI announcement, July 2025. URL https://x.ai/news/grok-4. Introduces Grok 4 Heavy with parallel test-time compute.

xAI. Multi agent. Grok API documentation, 2026. URL https://docs.x.ai/developers/model-capabilities/text/multi-agent. grok-4.20-multi-agent: 4 or 16 agents with a leader agent that synthesizes; released February 2026; accessed 2026-09-23.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2309.17453.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2412.06464.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. MemAgent: Reshaping long-context LLM with multi-conv RL-based memory agent. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2507.02259.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, et al. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2503.14476.

Alex L. Zhang, Tim Kraska, and Omar Khattab. Recursive language models, 2025. URL https://arxiv.org/abs/2512.24601. v1 December 2025; revised May 2026 with the SFT-trained RLM-Qwen3-8B.

Yusen Zhang, Ansong Ni, Ziming Mao, Chen Henry Wu, Chenguang Zhu, Budhaditya Deb, Ahmed H. Awadallah, Dragomir Radev, and Rui Zhang. Summˆ N: A multi-stage summarization framework for long input dialogues and documents. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022. URL https://arxiv.org/abs/2110.10150.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Ö. Arık. Chain of agents: Large language models collaborating on long-context tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2406.02818.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2306.14048.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization, 2025. URL https://arxiv.org/abs/2507.18071.

Zheqing Zhu, Christopher Wu, Yi Wan, Yiqing Cai, Qiwei He, Ruihao Zhu, Chang Zen Tee, Lexin Chen, Pu Thavikulwat, and Jared Ee. Pokee-Isaac 28B: A 10M-token context efficient agentic model. Technical report, Pokee AI, August 2026. URL https://console.pokee.ai/pokee-isaac-28b-v0-technical-report.pdf.

A HARNESS AND PROMPTS

The full system prompt and tool descriptions are given in Figure 5. The full-document prompt, used for single shot baselines, is given in Figure 6. Benchmark text fields are supplied in Table 2. Harness responses are given in Table 3.

You are a long-document assistant. The document is ⟨N⟩ tokens long; you cannot see it directly. Your own context window is ⟨B⟩ tokens — the conversation (system prompt, user message, your responses, and all tool results) must fit in this budget or the episode ends.

⟨*task context, if the benchmark provides one*⟩

You have two tools:
- `read_chunk(start, end)`: read the document tokens in [start, end).
- `spawn_subagent(subtask)`: delegate to a fresh-context copy of yourself (also ⟨B⟩ tokens) with `subtask` as the user prompt. The same tools and system prompt as above are inherited by the subagent.

When you are confident in the final answer, emit it as \boxed{value} and stop.

Tool descriptions (function-calling schema)

read_chunk: Read a slice of the document and return the decoded text of tokens [start, end). The document has a fixed length (stated in the system prompt). Each call is capped at the chunk limit; for larger ranges, issue multiple reads or delegate to a subagent.
start: First token position to read (inclusive). end: Last token position to read (exclusive).

spawn_subagent: Spawn a fresh-context copy of yourself to solve `subtask`; returns the child's \boxed{} answer.
subtask: Sub-problem statement (free text) for the child to solve

Figure 5: **Agent system prompt**, shared by the root and every subagent, and the two tool descriptions. ⟨N⟩ is the document length and ⟨B⟩ the per-agent context budget. The root's user message is the benchmark question (Table 2); a subagent's user message is the `subtask` string its parent wrote.

System

⟨*task context, if the benchmark provides one*⟩

You are given a complete document followed by a question about it. Read the document carefully and answer the question. Show your reasoning, then emit your final answer as \boxed{value} and stop.

User

⟨*full document text*⟩

⟨*question*⟩

Figure 6: **Full-document prompt**, used for the single-shot baselines: the whole document is placed in one user message, with no tools and no budget prose. For RULER, whose template embeds the document mid-prompt, the document is instead placed back at the template's {context} position, reproducing RULER's own prompt.

Benchmark	System prompt task context	Root user message
OOLONG-synth	OOLONG's dataset description, verbatim (its intro, the number of items, the label set)	OOLONG's question, verbatim, + "Put your final answer inside \boxed{}."
RULER	None (empty)	RULER's task template, verbatim, with {context} replaced by a pointer to the document (below) + "Reply with the answer inside \boxed{}."

Table 2: **Benchmark text fields.** The task context is placed in the system prompt, so every subagent inherits it without the parent restating it. For RULER, {context} in the template is replaced by: "[The relevant text is in a separate document accessible via the read_chunk tool — see the system prompt for usage.] (Document length: $\langle N \rangle$ tokens.)"

Event	Tool result returned to the calling agent (verbatim)
spawn_subagent	
Subagent ends with a boxed answer	The content of its last closed \boxed{...}
Subagent exceeds its context budget	`agent overflowed context`
Subagent stops without a boxed answer	`agent did not box an answer`
Tree node cap reached	`Error: rollout node budget reached. Solve directly.`
read_chunk	
Valid range	The decoded text of tokens `[start, end)`, clamped to the document
Empty range after clamping	`Empty range.`
Non-integer arguments	`Error: read_chunk requires integer 'start' and 'end'.`

Table 3: **Harness responses.** An agent's budget counts its whole rendered conversation (system prompt, tool schema, user message, its own turns and every tool result); each generation is capped at the remaining room. A subagent that answers returns only its boxed content, so its reads and reasoning never enter the parent's context.

B TRAINING DATA

A high-level description of our training tasks is given in Table 4.

Task	Strategy	N	Example question / what the leaf does
Order-independent aggregation over structured records (one record per line)			
`synth_sum`	tree	10	SUM of `amt` over all records
`synth_count`	tree	10	how many records have `flag=Y`
`synth_max`	tree	10	largest `amt`
`synth_min`	tree	10	smallest `amt`
`synth_sumwhere`	tree	10	SUM of `amt` over only the `flag=Y` records
`synth_maxwhere`	tree	10	largest `amt` among a flag-filtered subset
`synth_mode`	tree	10	most common `grp` (per-`grp` tally)
`synth_distinct`	tree	10	how many distinct `grp` values (set state)
`synth_sumby`	tree	10	which `grp` has the largest total `amt`
`synth_count2`	tree	10	count under a conjunction of two field conditions
`synth_diff`	tree	10	SUM over `flag=Y` minus SUM over `flag=N`
`synth_count_cmp`	tree	10	how many records have `amt` below a threshold
`synth_count_range`	tree	10	how many records have `amt` in an interval
`synth_filter_argmax`	tree	15	most / least common value among a filtered subset
`synth_2d`	tree	24	two-level questions (e.g. for how many months is `K3` the most common group)
Sequential folds			
`synth_runreset`	fold	20	running total that resets at `grp=RST`
`synth_varchain`	fold	20	final value of a variable under `set B = A` copies
`synth_peak`	fold	20	highest value the running total reaches
`synth_streak`	fold	20	longest run of consecutive `flag=Y` records
`synth_adjacent`	fold	20	records whose `amt` exceeds the previous record's
`synth_first_exceed`	fold	20	index where the running total first exceeds a threshold
`vt_novel`	fold	150	variable-assignment chains hidden in prose
Open-vocabulary frequency			
`synth_topk`	tree	120	the k most frequent words in a word list; partials are pruned `word:count` tallies
Multi-line records			
`long_records`	tree	120	header + prose-body entries; counts, argmaxes, 2-D tallies, and word mentions in bodies
Labelling real text, then aggregating			
`labeled_records`	tree	240	items from 7 labelled datasets; the leaf judges each label, then counts / compares
`rule_label`	tree	40	prose sentences labelled by a stated rule (e.g. ends with "?"), then aggregated
`realdoc_count`	tree	40	occurrences of a word in a Gutenberg novel passage
Retrieval and QA in prose			
`niah_novel`	tree	60	find the one sentence stating a key's magic value
`niah_multi`	tree	60	collect several hidden facts, or a lookup instruction plus all facts, and resolve at the root
`niah_bridge`	tree	80	two-hop: fact 1 names who holds a role, fact 2 answers about that person
`narrativeqa`	tree	30	short-answer questions over book / script windows; model-executed leaf
Total		1,229	

Table 4: **Training tasks.** N is the number of problems per task. *Tree*: recursive halving, with each parent merging its two children's partials (for retrieval, the half that finds the answer wins). *Fold*: one agent reads a slice and hands the rest of the document and its accumulator to the next. Every agent in each tree is a training example.

We upweight root gradient mass 4x and positive `narrativeqa` leaves 2x, downsampling negative leaves by half. Each task has prompt jitter to train a wider distribution. We also jitter the system prompt. We train with context budgets 3K (50%), 5K (15%), 8K (13%), 10K (11%), and 12K (11%), and document length between 4K and 14K tokens, with a distribution so document length is typically larger than context budget.

C TRAINING DETAILS

Our training runs used rank 32 LoRA on Qwen3.6-35B-A3B, chat renderer `qwen3_5_disable_thinking`, cross-entropy loss on assistant tokens only. Small batch training of size 8-64 (you pay per token on Tinker, and small batch training is generally better-behaved per token, particularly on LoRAs (Marek et al., 2025; Schulman & Thinking Machines Lab, 2025; Masters & Luschi, 2018; Keskar et al., 2017)). We used pure Adam optimizer, $\beta_1 = 0.9$, $\beta_2 = 0.95$, with constant learning rate between 5e-6 and 1e-5, single epoch training.