Self-Compacting Language Model Agents

Tianjian Li[♠] **Jingyu Zhang**[♠] **William Jurayj**[♠] **Xi Wang**[♠] **Chuanyang Jin**[♠]
Mehrdad Farajtabar[♡] **Eric Nalisnick**[♠] **Daniel Khashabi**[♠]

[♠]Johns Hopkins University [♡]Apple
{tli104, danielk}@jhu.edu

Abstract

Long agent traces composed of chains of thought and tool calls accumulate stale content that anchor subsequent generations, and eventually outgrow the context window. Existing scaffolds mitigate it with *fixed-interval* compaction triggered at a token threshold. Such triggers pay no heed to trajectory structure, risking discard of partial results mid-derivation or mid-search. We propose SELFCOMPACT, a scaffold that allows the model itself to decide *when* and *how* to compact. Specifically, it pairs two inference-time elements: (i) a *compaction tool* the model invokes to summarize the accumulated context, and (ii) a lightweight *rubric* specifying when to fire (a sub-task has resolved, or the trajectory is converging) and when to suppress (mid-derivation, or when stuck). Both are needed. The tool alone is unevenly used across open-weight models, often invoked at unhelpful moments or not at all; the rubric alone cannot act. Together, they elicit effective adaptive compaction without any fine-tuning or external supervision. We present empirical results on six benchmarks (competitive math and agentic search) and seven models. Our results show that SELFCOMPACT matches or exceeds fixed-interval summarization at a fraction of the token cost, improving over a no-summarization baseline by up to 18.1 points on math and 5–9 points on agentic search at 30–70% lower per-question cost. Our results expose a meta-cognitive gap: although unprompted models cannot reliably tell when their own context is rotting, a lightweight rubric closes this gap, reframing when to compact as a capability that scaffolds can supply without training[1].

1 Introduction

We are chasing after harder problems over longer horizons [METR, 2026], and consequently, the trajectories LMs generate to solve them keep growing. Reasoning models are now able to spend tens of thousands of tokens deliberating on a single competition math question: Qwen3.5 produces 81k tokens [Qwen Team, 2026], Kimi-K2.5 produces 96k [Kimi Team et al., 2026]. Agentic systems extend further, orchestrating search results [Wei et al., 2025], code execution outputs [Jimenez et al., 2024], and intermediate plans [Novikov et al., 2025] across hundreds of turns. The bet that more thinking and interactions yields better answers has paid off, but long traces carry a hidden cost.

As the trace grows, it accumulates junk — a flawed case split made early, a search result the model has moved past, a candidate program that led nowhere. These leftovers do not just sit there; they anchor everything that follows [Laban et al., 2026]. A model that solves a problem from a clean start often fails when fed back its own flawed reasoning. This phenomenon is known as *context rot* [Hong et al., 2025, Cheng et al., 2026]. Existing systems try to manage it with rigid rules: compacting the trajectory when a token threshold is met [Cursor Research et al., 2026], or delegating the burden of identifying context rot to the user via `/compact` [Anthropic, 2025].

[1]Code is available at https://github.com/tianjianl/selfcompact.

Preprint. Under Review.

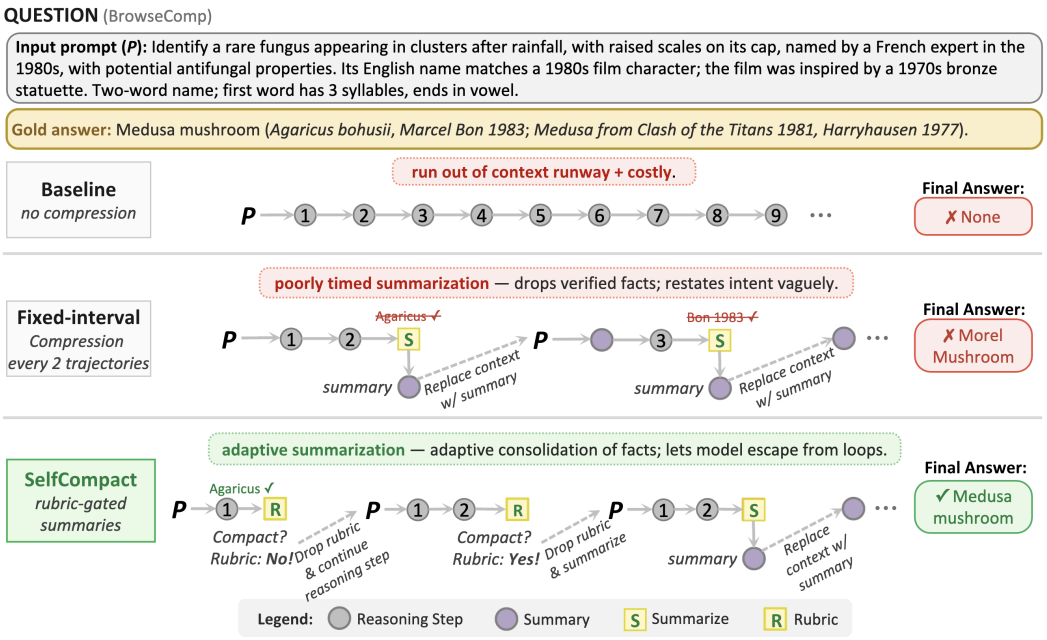

Figure 1: **Comparison of trajectory-compression strategies on a hard BrowseComp question.** The gold answer requires verifying four facts (*Agaricus*, *Bon 1983*, *Clash 1981*, *Harryhausen*) before composing *Medusa mushroom*. **Baseline** (no compression) burns its budget on an unproductive monologue and emits no answer. **Fixed-interval** compression fires every two search trajectories regardless of reasoning state; the poorly-timed summary wipes verified facts mid-reasoning, leaving the model to guess (*Morel Mushroom*). **SELFCOMPACT (Ours)** gates compression on a rubric over closed reasoning units, so each summary fires immediately after a verified fact—all four facts persist and the model returns the correct answer.

Today, this rigidity is the rule. Industrial deployed systems either wait until fixed token intervals to invoke compaction [Cursor Research et al., 2026]; recent academic scaffolds that compact context on reasoning traces [Wu et al., 2026a, Yan et al., 2026] and deep research systems [Chen et al., 2026], all rely on a token threshold to trigger compaction. For example, Liu et al. [2025] triggers compaction when "token usage exceeds 30% of the maximum context."[2] But a fixed threshold knows nothing about what the model is doing when it invokes summarization. It cannot distinguish a model mid-derivation from one that has just resolved a sub-problem, mid-search from one converging on an answer. The cost is asymmetric: a well-timed call discards stale work, but a poorly timed call discards the partial result the model needed to continue. These scaffolds help, but they leave substantial headroom. The right time to compact heavily depends on what the model is doing, not just on how many tokens have passed. Figure 1 shows this concretely: fixed-interval compression wipes four already-verified facts mid-trajectory, and the model falls back to a guess. In this work, we ask the following research question:

Can the LM agent itself recognize *its own context rot without training, and compact accordingly?*

To answer the research question, we hand the decision of when to compact back to the LM agent itself. We propose SELFCOMPACT, a scaffold that pairs two inference-time elements: an inline *compaction tool* the model invokes itself, and a lightweight *rubric* specifying when to fire (a sub-task has resolved, or the trajectory is converging) and when to suppress (mid-derivation, or when stuck). The model emits `<summarize>` when it wants to compact, the scaffold runs the summarization, and generation resumes from the summarized prefix. Both the rubric judge and summarizer is the same model: no external verifier, no fine-tuning.

[2] https://huggingface.co/MiniMaxAI/MiniMax-M2.5

Both elements are needed. Across seven open-weight models, four `Qwen` models on competition math and three deployed search agents (`MiniMax-M2.5`, `GLM-4.7-Flash`, `MiMo-V2-Flash`), ablating the rubric exposes uneven tool use: some models call the tool at unhelpful moments, others not at all. With the rubric in place, SELFCOMPACT elicits effective adaptive compaction without any fine-tuning, matching or exceeding fixed-interval summarization at a fraction of the token cost: on competition math (IMO-Answerbench, HMMT Nov 25 / Feb 26), it improves over a no-compaction baseline by up to 18.1 points; on agentic search (BrowseComp, BrowseComp-Plus, DeepSearch QA), it adds 5–9 points at 30–70% lower per-question cost.

To sum up, our contribution is three-fold:

- We propose SELFCOMPACT, a novel scaffold that exposes summarization as an inline tool paired with a lightweight rubric that tells the model when to invoke it. No fine-tuning required.
- Across seven open-weight models on competition math (four reasoning models) and agentic search (three deployed agents), SELFCOMPACT matches or exceeds fixed-interval summarization at substantially lower token cost.
- Ablations find that the rubric is crucial for effective self-compaction: offering the tool alone yields uneven behavior, while a paragraph of guidance closes the gap. Our findings imply that knowing when to perform compaction can be supplied as a meta-cognitive capability at inference time, rather than baked into weights.

To the best of our knowledge, our work is the first to introduce and evaluate rubric-based adaptive compaction timing in a training-free setting.

2 Preliminaries

Context management in long-horizon agents. LM agents on long-horizon tasks accumulate tokens in a single rolling context: prior thoughts, tool calls, and observations. A single coding-agent session on SWE-bench-style tasks routinely spans millions of tokens and well over a hundred turns. e.g., Qwen3-Coder-Next averages 8M tokens and 154 turns per problem on SWE-rebench [Badertdinov et al., 2026]. Deep-research agents on BrowseComp scale to hundreds of tool calls per query [Wei et al., 2025, MiroMind Team et al., 2026]. Even single-shot reasoning is no longer short: Kimi-K2.5 [Kimi Team et al., 2026] generates up to 96k thinking tokens on a single competition math problem. As horizons and task difficulty continue to scale, this accumulated context is not merely expensive but actively harmful: model performance degrades sharply with length, a phenomenon known as *context rot* [Hong et al., 2025]. Compaction is therefore a prerequisite for sustained long-horizon capability.

Current strategies. Today's agents compact via two content-agnostic heuristics. *Reactive* compaction triggers only when the rolling context approaches the model's token budget, treating compaction purely as overflow prevention [Anthropic, 2025]. *Periodic* compaction fires on a *fixed interval*—every k turns or every k tokens—regardless of what the trajectory contains [Cursor Research et al., 2026, Wu et al., 2026a]. Both ignore the state of the trajectory, and fail in opposite directions: reactive compaction waits until the context is already saturated with stale or erroneous tokens that have been polluting generation for many steps, while periodic compaction discards context indiscriminately, often summarizing in the middle of an active subgoal and erasing information the model still needs. Figure 1 illustrates how a fixed-interval summary firing mid-reasoning on a BrowseComp trajectory can wipe four already-verified facts, and the model is left to guess.

3 Our Approach: SELFCOMPACT

Setup. Given a natural language prompt x, a language model $\pi(\cdot \mid x)$ generates a continuation $y = (y_1, y_2, \ldots)$ autoregressively, with each token conditioned on the prompt and all preceding tokens. We equip the language model with a summarization tool $\mathcal{S} : (x, y_{1:t}) \to \tilde{y}$, which takes the original prompt x and a (possibly partial) continuation $y_{1:t}$ and produces a condensed version $\tilde{y}$. Generation then resumes from the new context $(x, \tilde{y})$. Crucially, the summarizer is the same model π—no external verifier or auxiliary model is required.

Algorithm 1: SELFCOMPACT: rubric-gated inference-time context compaction.

Input: prompt x; model π; probe interval N; step budget T; rubric prompt P_R; summarizer prompt P_S.

Notation : $\mathcal{C} \leftarrow \mathcal{C} \circ m$ appends message m to context $\mathcal{C}$; $z \sim \pi(\cdot \mid \mathcal{C})$ samples response z from π given $\mathcal{C}$; $\text{pop}(\mathcal{C}, m)$ pop (removes) message m from $\mathcal{C}$ (assuming that it's the last item in $\mathcal{C}$).

$\mathcal{C} \leftarrow x$; `// Overall context and its KV cache`
for $t = 1, \ldots, T$ **do**
  $y_t \sim \pi(\cdot \mid \mathcal{C})$; $\mathcal{C} \leftarrow \mathcal{C} \circ y_t$;
  if y_t *is a final answer* **then return** y_t;
  if $t \bmod N = 0$ **then**
    $\mathcal{C} \leftarrow \mathcal{C} \circ P_R$; `// rubric probe; KV prefix reused`
    $r_t \sim \pi(\cdot \mid \mathcal{C})$, $r_t \in \{\text{COMPRESS}, \text{CONTINUE}\}$; $\mathcal{C} \leftarrow \mathcal{C} \circ r_t$;
    if $r_t = \text{COMPRESS}$ **then**
      $\mathcal{C} \leftarrow \mathcal{C} \circ P_S$; `// KV prefix still reused`
      $\tilde{y} \sim \pi(\cdot \mid \mathcal{C})$; `// summarize trajectory`
      $\mathcal{C} \leftarrow x \circ \tilde{y}$; `// hard reset; resume from summary`
    else
      $\text{pop}(\mathcal{C}, r_t)$; $\text{pop}(\mathcal{C}, P_R)$; `// revert; trajectory unchanged`
    end
  end
end
return y_T;

SELFCOMPACT. SELFCOMPACT pairs two inference-time elements: (i) the summarization tool $\mathcal{S}$ exposed to the model, and (ii) a lightweight *rubric* that decides, at periodic probe intervals, whether $\mathcal{S}$ should fire. Both elements are needed: exposing the tool alone—letting π emit `<summarize>` whenever it wants—yields uneven, often poorly-timed firing across models: some call the tool reflexively at unhelpful moments, others rarely at all. The rubric closes that gap by translating "a sub-task has resolved" and "the trajectory is not stuck mid-derivation" into concrete, cite-able conditions π can verify against the trajectory in front of it. SELFCOMPACT runs on a single inference engine with no fine-tuning.

Summarizer design. To maximize KV-cache reuse, we implement $\mathcal{S}$ by appending a summarizer prompt (Box A for Math; Box B for Search Agents) as a user-role message: the cache of $y_{1:t}$ at trigger time is preserved across the call, so $\mathcal{S}$ pays prefill only on the appended instruction, not on the full trajectory. A saving we quantify in the cost analysis below. The scaffold appends the instruction to $(x, y_{1:t})$, π generates the summary $\tilde{y}$, and $\tilde{y}$ then replaces $y_{1:t}$; decoding resumes from the original question and the summarized content $(x, \tilde{y})$.

Triggering the summarizer. At intervals $t \in \{N, 2N, 3N, \ldots\}$, the scaffold appends a rubric prompt P_R as a user-role message to $(x, y_{1:t})$ and samples from $\pi(x, y_{1:t}, P_R)$ a binary verdict $r_t \in \{\text{COMPRESS}, \text{CONTINUE}\}$. These intervals are measured in tokens of the reasoning trace (math) or in tool calls (agentic search).

As with $\mathcal{S}$, the probe is appended rather than substituted so that the KV cache of $y_{1:t}$ is preserved across the call, keeping the judge near-free relative to the trajectory length. The probe enumerates a small set of conditions, each requiring verbatim evidence quoted from the trajectory, so π can check them locally. The rubric is task-specific: the conditions for competition math differ from those for agentic search, and we defer the exact wording to Appendix A (math) and Appendix B (search). On a COMPRESS verdict, the summarizer fires:

$$\tilde{y} \leftarrow \mathcal{S}(x, y_{1:t}) \quad \text{whenever } t \equiv 0 \pmod{N} \text{ and } r_t = \text{COMPRESS};$$

otherwise generation resumes from $(x, y_{1:t})$ unchanged.

Workflow. Our experiments compare three regimes that share the same generation loop but differ in how the transcript is compressed: **no compaction** (the trajectory accumulates untruncated),

fixed-interval compaction (summarize-and-replace at every probe interval, and SELFCOMPACT (summarize-and-replace only when the rubric returns COMPRESS, otherwise continue). On agentic search this yields a *search–judge–summarize–search* loop, with the agent continuing against $\tilde{y}$ rather than the full history after each compression.

Cost analysis. Define $L = |y_{1:t}|$ as the pre-compression trajectory length and $\ell = |\tilde{y}|$ for the summary length, both in tokens. SELFCOMPACT adds at most two LLM calls per probe interval: the rubric probe always, the summarizer only on a COMPRESS verdict. Both append to $(x, y_{1:t})$, so the KV cache of $y_{1:t}$ is reused across the call and the $\mathcal{O}(L^2)$ re-prefill that a naive re-encode would incur is avoided. The rubric probe is essentially free: a short verdict r_t generated on top of the cached prefix. The summarizer is the only real overhead: $\mathcal{O}(L\ell)$ to generate $\tilde{y}$ plus a one-time $\mathcal{O}(\ell^2)$ prefill of the new $(x, \tilde{y})$ prefix. Empirically the trade is favorable: on agentic search SELFCOMPACT adds 5–9 accuracy points at 30–70% lower per-question token cost than the no-compaction baseline (§4.2); on competition math it leads on 11 of 12 benchmark/model cells under a token budget matched to fixed-interval summarization (§4.1). Appendix C states the per-question cost formula in full, defines how $(N_{\text{prefill}}, N_{\text{cached}}, N_{\text{out}})$ are accumulated from each trajectory, lists the OpenRouter prices, and works out the cached-vs-prefill break-even (compaction wins iff $L/\ell > 10$, which the search summarizer reaches at 20–80×).

4 Experiments

4.1 Multi-turn Reasoning in Competition Math

Setup. We evaluate 4 models of different sizes in the Qwen family — 2 instruct models `Qwen3-4B-Instruct-2507` and `Qwen3-30B-A3B-Instruct-2507`, and 2 Qwen3.5 [Qwen Team, 2026]: `Qwen3.5-4B` and `Qwen3.5-9B`, on 3 benchmarks: IMO-Answerbench [Luong et al., 2025], HMMT Nov 2025 and HMMT Feb 2026 [Balunović et al., 2025]. We generate 16 responses for each question in parallel and report the mean across samples. We use vLLM [Kwon et al., 2023] for inference and the `math_verify` package for parsing and evaluation.

We compare SELFCOMPACT against fixed interval summary, where summarization is always triggered for every 16k tokens, we keep making the model compact until it uses a similar amount of tokens compared to SELFCOMPACT for a fair comparison. We use the same scaffold as ReasoningCache [Wu et al., 2026a], prompting the model to continue generating based on an existing trajectory or a summary of it. Detailed summarization and continuation prompts can be found in Appendix A.

Results. Table 1 shows the results across three competition math benchmarks and four model configurations. Under matched token budgets, SELFCOMPACT consistently outperforms both the baseline and fixed interval summarization, achieving the best performance in 11 out of 12 settings. The gains are most pronounced for the thinking-enabled models: on Qwen3.5-9B, SELFCOMPACT improves over the baseline by 16.4, 10.0, and 18.1 points on IMO-Answerbench, HMMT Nov, and HMMT Feb respectively. The one exception is `Qwen3-30B-A3B` on HMMT Feb, where fixed interval summarization edges out SELFCOMPACT by 1.1 points, though the latter still leads on the other two benchmarks.

Headroom analysis. We analyze the failure modes of fixed interval summarization for `Qwen3-4B-Instruct-2507` on IMO-Answerbench by tracking answer transitions across the 12 summarization calls. Table 2 reports the number of instances where the model's answer changes from correct to wrong after a summarization and vice versa.

Table 2: Answer transitions across 12 fixed interval summarization calls for Qwen3-4B-Instruct-2507 on IMO-Answerbench.

Transition	Count
Wrong → Correct	1486
Correct → Wrong	1009

While summarization produces a net gain, 40.4% of all transitions cause degradations. This motivates a natural question: what if the model could selectively skip summarization when its current answer is already correct? Table 3 shows that an oracle policy which suppresses summarization calls whenever the current answer is correct but otherwise follows the same fixed schedule achieves 52.9%, a +11.5 improvement over fixed interval and +14.0

Table 1: Performance across competition math benchmarks, for fixed interval summary, we constrain the avg. number of tokens per question as the same as SELFCOMPACT. Numbers in brackets indicate the average number of tokens used per question.

	IMOBench	$\text{HMMT}_{\text{Nov 25}}$	$\text{HMMT}_{\text{Feb 26}}$	Avg.
`Qwen3-4B-Instruct-2507`				
No Compaction [16k]	$38.9_{1.7}$	$40.8_{2.1}$	$36.5_{2.9}$	$38.7_{2.2}$
Fixed Interval Summary [44k]	$41.4_{2.1}$	$44.0_{2.7}$	$39.2_{1.8}$	$41.5_{2.2}$
SELFCOMPACT [48k]	$\mathbf{45.5}_{2.8}$	$\mathbf{47.8}_{1.8}$	$\mathbf{42.1}_{2.8}$	$\mathbf{45.1}_{2.5}$
`Qwen3-30B-A3B-Instruct-2507`				
No Compaction [16k]	$45.2_{1.2}$	$54.7_{2.7}$	$51.9_{2.8}$	$50.6_{2.2}$
Fixed Interval Summary [26k]	$48.7_{2.9}$	$57.3_{1.7}$	$\mathbf{58.7}_{1.9}$	$54.9_{2.2}$
SELFCOMPACT [29k]	$\mathbf{52.1}_{2.6}$	$\mathbf{59.4}_{1.2}$	$57.6_{1.4}$	$\mathbf{56.4}_{1.7}$
`Qwen3.5-9B (Thinking Disabled)`				
No Compaction [16k]	$25.0_{1.0}$	$38.2_{1.2}$	$34.2_{2.7}$	$32.5_{1.6}$
Fixed Interval Summary [90k]	$33.4_{2.7}$	$42.1_{2.2}$	$44.9_{2.9}$	$40.1_{2.6}$
SELFCOMPACT [93k]	$\mathbf{41.4}_{2.3}$	$\mathbf{48.2}_{1.6}$	$\mathbf{52.3}_{2.0}$	$\mathbf{47.3}_{2.0}$
`Qwen3.5-4B (Thinking Disabled)`				
No Compaction [16k]	$16.9_{3.0}$	$26.4_{1.2}$	$22.5_{1.6}$	$21.9_{1.9}$
Fixed Interval Summary [64k]	$27.4_{0.9}$	$35.5_{2.0}$	$29.1_{2.9}$	$30.7_{1.9}$
SELFCOMPACT [67k]	$\mathbf{34.1}_{2.7}$	$\mathbf{37.4}_{1.1}$	$\mathbf{30.0}_{2.2}$	$\mathbf{33.8}_{2.0}$

over baseline. Notably, this oracle is a strict subset of what a fully adaptive policy could achieve, as it only decides *whether* to summarize at each fixed interval, not *when* to summarize.

Both our results on Qwen3.5, where adaptive summarization already outperforms fixed interval, and this oracle analysis indicate substantial headroom for adaptive policies that learn when summarization is beneficial.

Table 3: Oracle analysis on IMO-Answerbench for Qwen3-4B-Instruct-2507.

Method	Accuracy (%)
No Compaction [16k]	$38.9_{1.7}$
Fixed Interval Summary [44k]	$41.4_{2.1}$
SELFCOMPACT [48k]	$45.5_{2.8}$
Oracle (skip if correct) [44k]	$52.9_{1.4}$

4.2 Agentic Search

Setup. We evaluate 3 models of different sizes and families: `GLM-4.7-Flash` (30B total, 3B active; [GLM Team et al., 2025]), `MiniMax-M2.5` (230B total, 10B active; [MiniMax Team, 2026]) and `Mimo-V2-Flash` (309B total, 15B active; [MiMo Team et al., 2026]) on 3 agentic search tasks: BrowseComp [Wei et al., 2025] , BrowseComp plus [Chen et al., 2025] and DeepSearch QA [Gupta et al., 2026]. We use the exact same scaffold and evaluation settings as Lee et al. [2026]. We defer detailed sampling parameters and prompts to Appendix B.

For cost, we report the per-question USD cost. In the main text we use a single-rate approximation that bills every prompt token at the cached rate, $\text{COST}(q) = (p_{\text{cache}}\, N_{\text{prompt}} + p_{\text{out}}\, N_{\text{out}})/10^6$, where N_{prompt} and N_{out} are the cumulative prompt and completion tokens summed across every LLM call in the trajectory (assistant turns, rubric probes, and summaries). This is conservative: the first-time prefill tokens that a provider bills at the higher uncached rate p_{in} are only a small fraction (empirically 5–8%) of N_{prompt}, so the effective rate stays close to p_{cache}. Prices come from OpenRouter (Table 8). Appendix C gives the full procedure, and Appendix D reports the finer-grained accounting that separates first-time prefill (billed at p_{in}) from cached reads (billed at p_{cache}). We compare SELFCOMPACT against the following baseline methods:

- **No Compaction**: No context management is used, model stops when either max context window is reached, or max number of tool calls are executed.
- **Fixed-interval summary**: Summarization is triggered when 30% of the max context window is consumed, a threshold suggested by Liu et al. [2025], closely resembling Wu et al. [2026a].
- **Delete-all**: All the past trajectory is discarded when 30% of the max context window is consumed.

- **Keep-last-N**: The last 3 turns are saved and the rest is discarded when 30% of the max context window is consumed.

Main Results. Table 4 shows the results of the three deployed agents (`GLM-4.7-Flash`, `MiniMax-M2.5`, `MiMo-V2-Flash`) on agentic search. SELFCOMPACT is the strongest method for all three models, improving over the no-compaction baseline by $+8.5$ (GLM-4.7-Flash), $+9.2$ (MiniMax-M2.5), and $+5.3$ (Mimo-V2-Flash) absolute points on BrowseComp-Plus.

The accuracy ordering is consistent across all three models: baseline $<$ fixed-interval $\leq$ SELFCOMPACT. Fixed-interval summarization recovers most of the baseline's lost ground; SELFCOMPACT's autonomous trigger adds up to $+6.3$ points by avoiding poorly-timed compressions the rubric concentrates that distribution at meaningful stopping points.

On the cost axis, SELFCOMPACT is *cheaper* than the baseline despite issuing an extra LLM call at each probe. Per-question cost on BrowseComp-Plus drops by 67% (GLM-4.7-Flash, $0.12 \rightarrow 0.04$), 63% (MiniMax-M2.5, $0.19 \rightarrow 0.07$), and 33% (Mimo-V2-Flash, $0.24 \rightarrow 0.16$). The KV-cache reuse described in §3 is what makes the probe affordable: each rubric judgement contributes only its own short generation, since the trajectory's cache is preserved. The bulk of the savings come from the post-summarization regime, where every subsequent token attends to $\tilde{y}$ rather than $y_{1:t}$.

Table 4: Performance and per-question cost across benchmarks. "Cost" is USD per question, computed with a single-rate approximation that bills every prompt token at the cached rate, $(p_{\text{cache}}\, N_{\text{prompt}} + p_{\text{out}}\, N_{\text{out}})/10^6$, where N_{prompt} and N_{out} are the cumulative prompt and completion tokens summed across every LLM call in the trajectory; Appendix D reports the finer-grained cost that separates first-time prefill from cached reads, and the full bookkeeping is in Appendix C. We highlight the cost reduction % compared to the "no compaction" baseline for SELFCOMPACT. **Bold** indicates the best value per column within each model block. Following Sun et al. [2025] and Lee et al. [2026], evaluations are done on 150 subsampled questions per benchmark.

	Context Management	BrowseComp		BrowseComp-Plus		DeepSearchQA		Overall	
		Acc. ↑	Cost ↓	Acc. ↑	Cost ↓	Acc. ↑	Cost ↓	Acc. ↑	Cost ↓
`GLM-4.7-Flash`	No Compaction	$30.1_{1.0}$	0.14	$45.6_{0.7}$	0.12	$34.2_{0.3}$	0.13	$36.6_{0.4}$	0.13
	Fixed-interval	$35.4_{1.8}$	0.06	$50.0_{2.3}$	0.04	$39.1_{0.3}$	0.05	$41.5_{1.0}$	0.05
	Delete-all	$31.6_{1.3}$	**0.03**	$47.3_{1.3}$	**0.02**	$35.8_{1.5}$	**0.03**	$38.2_{0.8}$	**0.03**
	Keep-last-N	$33.5_{2.4}$	0.05	$51.4_{1.6}$	0.03	$38.8_{0.5}$	0.04	$41.2_{1.0}$	0.04
	SELFCOMPACT (Ours)	**41.2**$_{2.3}$	0.10 (-29%)	**54.1**$_{1.4}$	0.04 (-67%)	**44.0**$_{1.3}$	0.07 (-46%)	**46.4**$_{1.0}$	0.07 (-46%)
`MiniMax-M2.5`	No Compaction	$47.2_{0.7}$	0.19	$62.0_{1.8}$	0.19	$52.0_{1.4}$	0.19	$54.6_{0.8}$	0.19
	Fixed-interval	$55.3_{1.6}$	0.07	$65.9_{2.1}$	0.04	$56.7_{0.6}$	0.06	$59.3_{0.9}$	0.06
	Delete-all	$52.7_{1.3}$	**0.05**	$65.0_{1.3}$	**0.03**	$54.9_{0.7}$	**0.04**	$57.5_{0.7}$	**0.04**
	Keep-last-N	$54.5_{1.7}$	0.06	$67.4_{0.4}$	0.05	$57.0_{1.8}$	0.06	$59.6_{0.8}$	0.06
	SELFCOMPACT (Ours)	**59.3**$_{1.4}$	0.09 (-53%)	**71.2**$_{2.6}$	0.07 (-63%)	**61.3**$_{2.2}$	0.08 (-58%)	**63.9**$_{1.2}$	0.08 (-58%)
`Mimo-V2-Flash`	No Compaction	$42.7_{0.9}$	0.27	$57.6_{1.4}$	0.24	$46.5_{1.4}$	0.25	$48.9_{0.7}$	0.25
	Fixed-interval	$51.6_{2.2}$	0.14	$60.2_{1.9}$	0.14	$52.3_{1.4}$	0.14	$54.7_{1.1}$	0.14
	Delete-all	$45.5_{1.1}$	0.08	$61.2_{2.3}$	**0.11**	$49.7_{2.1}$	**0.09**	$52.1_{1.1}$	**0.09**
	Keep-last-N	$49.8_{0.8}$	**0.06**	**63.5**$_{1.7}$	0.15	$53.0_{0.5}$	0.10	$55.4_{0.6}$	0.10
	SELFCOMPACT (Ours)	**57.9**$_{2.4}$	0.11 (-59%)	$62.9_{2.2}$	0.16 (-33%)	**56.8**$_{2.2}$	0.13 (-48%)	**59.2**$_{1.3}$	0.13 (-48%)

SELFCOMPACT fires compression earlier. We plot the number of tokens accumulated before each compaction fires in Figure 2, comparing Fixed-Interval (30% of context) against SELFCOMPACT on BrowseComp-Plus. SELFCOMPACT's rubric-fired summaries skew to the left of the 30% threshold line in all three models, whereas Fixed-Interval pins triggers at the threshold by construction. This left skew indicates that the rubric reaches its "compress now" verdict well before the trajectory consumes 30% of context. This indicates the fixed threshold typically fires too late, retaining stale tokens past the point where the model has already resolved the current sub-question.

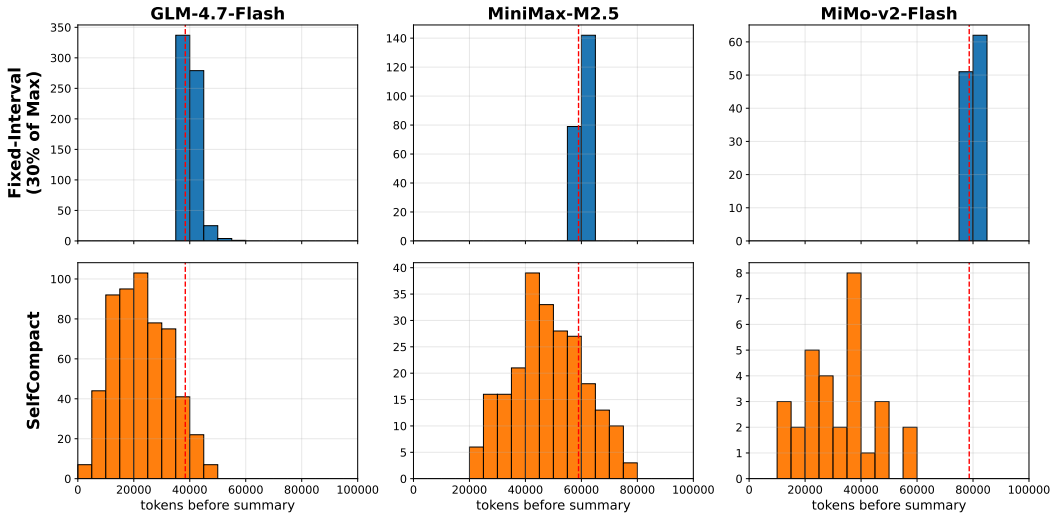

Figure 2: Distribution of context length when a summary fires in BrowseComp Plus. Top: Fixed-Interval: the policy compresses at a hard 30%-of-max-context budget (red dashed). Bottom: SELF-COMPACT — the rubric fires when the model judges a sub-question resolved. Fixed-Interval triggers compaction at the threshold by construction; SELFCOMPACT spreads them across the budget.

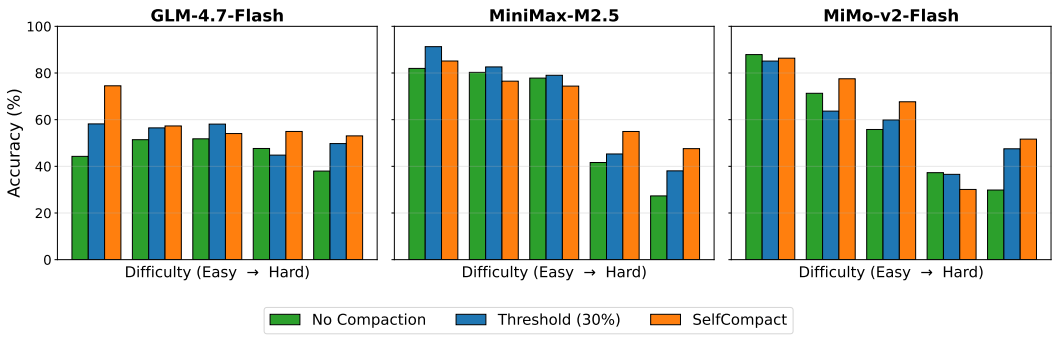

Figure 3: Accuracy by per-question difficulty on BrowseComp Plus for three models. Difficulty is the total output tokens consumed by the no-compression baseline, split into 5 equal-size quantile bins per model (left = easy, right = hard). Bars compare Baseline (no compression), a fixed-budget Threshold (30% of context), and SELFCOMPACT. SELFCOMPACT matches the others on easy bins and pulls ahead on the hardest bins across all three models.

SELFCOMPACT helps on harder problems. We bin each question by the total output tokens its no-compression baseline trajectory consumed (a model-specific proxy for question difficulty) and split into five equal-size quantiles per model. Within each bin we report accuracy for the baseline, a fixed-budget Threshold (30% of context), and SELFCOMPACT (Figure 3). On easy bins the three policies are within sampling noise, but on the two hardest bins SELFCOMPACT improves over the Threshold baseline by 5–20 pp consistently across 3 models, indicating that rubric-driven compression helps the most on questions that require deep search and accumulate the largest contexts.

Ablation on the effect of rubrics. We ablate the effect of rubrics by removing the rubric-gated verification and instead letting the model self-decide whether to summarize at each step. This collapses performance to 41.0% on average with `GLM-4.7-Flash` (Table 5), a 5.4-point drop from full SelfCompact (46.4%) that leaves it on par with the naive fixed-interval schedule (41.5%) and below it on BrowseComp (33.6 vs 35.4), with the same effect on math reasoning using `Qwen3-4B-Instruct-2507` (40.9% vs 45.5%, Table 6). This shows that the gains do not come from triggering compaction per se, but from the rubric-gated verification that determines what reasoning state is safe to preserve; without it, the model's unconstrained summarization discards information

Table 5: Accuracy (%) across agentic search benchmarks with `GLM-4.7-Flash`. Subscripts denote standard deviation over 5 runs. Best per column in bold.

Method	**BrowseComp**	**BrowseComp-Plus**	**DeepSearchQA**	**Avg.**
No Compaction	$30.1_{1.0}$	$45.6_{0.7}$	$34.2_{0.3}$	$36.6_{0.4}$
Fixed-interval	$35.4_{1.8}$	$50.0_{2.3}$	$39.1_{0.3}$	$41.5_{1.0}$
SELFCOMPACT w/o rubrics	$33.6_{2.1}$	$51.9_{2.4}$	$37.6_{1.9}$	$41.0_{1.2}$
SELFCOMPACT	$\mathbf{41.2}_{2.3}$	$\mathbf{54.1}_{1.4}$	$\mathbf{44.0}_{1.3}$	$\mathbf{46.4}_{1.0}$

that later steps depend on. In other words, knowing when to compact is insufficient, and the rubrics are what make the retained context reliable.

5 Related Work

Context compaction in frontier models. In API-based models, context compaction has become a standard built-in feature to prevent multi-rounds of long reasoning chains from blowing up the context window.[3] Note that while the feature is often named as "auto compaction", it refers to the compaction operation being automatically triggered and performed when the *context length reaches a target threshold*, i.e. fixed interval. Such mechanisms have raised complaints from the community[4], where compaction at inappropriate positions tends to remove important information from the context, which is a central issue we aim to resolve. The only exception is a new feature from LangChain posted concurrently to our work[5] March 2026, where they also introduced a summary tool; they did not introduce the rubric-based criteria for deciding *when* to summarize.

Table 6: Accuracy (%) on IMOBench with `Qwen3-4B-Instruct-2507`. Subscripts denote standard deviation.

Method	Math
No Compaction	$38.9_{1.7}$
Fixed-interval	$41.4_{2.1}$
SELFCOMPACT w/o rubrics	$40.9_{2.3}$
SELFCOMPACT	$\mathbf{45.5}_{2.8}$

Learning to compact during post-training. Long-horizon reasoning and agentic search accumulate tokens at an unsustainable rate: observations pile up, intermediate results go stale, and prior context anchors rather than informs subsequent generations. A broad line of work addresses this through periodic compaction, sharing a common paradigm: generate an extended sequence of thoughts or actions, summarize the accumulated context, resume from the compressed state, and repeat [Shao et al., 2025, Sun et al., 2025]. The majority of these methods *train* models to perform compaction (via SFT, RL, or both) in order to encode the when-to-compact decision implicitly in teacher demonstrations or rewards [Kontonis et al., 2026, Yang et al., 2026, Yan et al., 2026, Cassano and Rush, 2026, Mitra and Ulukus, 2025, Aghajohari et al., 2025, Wu et al., 2026a, Sun et al., 2025, Ye et al., 2025, Shao et al., 2025, Wu et al., 2026b]. Our approach is entirely training-free. We draw on recent advances in tool-using models [GLM Team et al., 2025, MiniMax Team, 2026], which have unlocked the ability to invoke external tools mid-generation, and show that off-the-shelf models can self-compact effectively if scaffolded correctly. A lightweight rubric specifying when to invoke the compaction tool is sufficient to elicit reliable adaptive compaction without any parameter updates.

Across these methods, most also trigger compaction at *fixed* token-count thresholds independent of trajectory structure [Yan et al., 2026, Aghajohari et al., 2025, Wu et al., 2026a,b], risking compaction mid-derivation when the model still needs the context it is about to discard. Our rubric is designed precisely to detect these structural conditions, firing when a sub-task has resolved or the trajectory is converging, and suppressing when the model is mid-derivation or stuck.

Few works have studied compaction without model training. The closest to ours is by Zhu et al. [2026a], which introduces a training-free variant that uses a frontier model as an external summarizer applied at *fixed* intervals after each full search trajectory. Our work differs in that, we introduce a rubric that dynamically determines when compaction should fire rather than relying on fixed post-

[3] e.g. `https://platform.claude.com/cookbook/tool-use-automatic-context-compaction`

[4] `https://www.reddit.com/r/codex/comments/1qib69i/auto_compaction_is_not_that_helpful/`

[5] `https://www.langchain.com/blog/autonomous-context-compression`

trajectory intervals. Our work also connects to a broader goal of context management. We refer the interested reader to the existing surveys on this topic [Mei et al., 2025].

KV Cache Eviction. A complementary line of work reduces memory and compute costs by evicting or compressing KV cache entries during inference. Static methods retain tokens based on recency [Xiao et al., 2023] or attention scores [Li et al., 2024], while more recent approaches use richer importance measures that are often learned [Park et al., 2025, Eyuboglu et al., 2025, Zweiger et al., 2026, Monea et al., 2025, Li et al., 2026]. Across this literature, interpretability of the compressed representation is treated as secondary: what is retained or evicted need not be human-readable, only attention-efficient, and may even be complemented by black-box weight updates [Zuo et al., 2026]. The context compaction literature (which is where our work fits in), by contrast, prioritizes discrete natural-language summaries that remain inspectable and overridden.

6 Limitations

Model capability gap. We only evaluated open-weight models; frontier systems (GPT-5.5 [OpenAI, 2026], Claude Opus 4.7 [Anthropic, 2026], Gemini 3 Pro [Google, 2026]) report stronger metacognition [Lindsey, 2025] and may detect context rot without a rubric. SELFCOMPACT is complementary: a training-free scaffold that layers onto any model and closes the gap precisely where it exists in deployed open-weight agents.

No reinforcement learning. We scope this work to training-free interventions to isolate the rubric's contribution. Prior work [Kontonis et al., 2026, Zhu et al., 2026b] shows RL can teach a model both *when* and *what* to compact. We view RL as a natural extension: the rubric supplies a behavioral target that RL can distill into the policy.

7 Conclusion

We introduced SELFCOMPACT, a rubric-gated summarization scaffold that compacts agent trajectories at closed reasoning units rather than on fixed intervals. Across six benchmarks and seven models, it matches or exceeds fixed-interval summarization at 30–70% lower cost, reframing *when to compact* as a capability that a lightweight rubric can supply without training.

Acknowledgments

This work was supported in part by Apple and Defense Advance Research Projects Agency (DARPA) under Contract No. HR001125C0304 and ONR grant (N0001424-1-2089). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA. We acknowledge the use of computational resources from the Johns Hopkins DSAI cluster.

References

Milad Aghajohari, Kamran Chitsaz, Amirhossein Kazemnejad, Sarath Chandar, Alessandro Sordoni, Aaron Courville, and Siva Reddy. The markovian thinker: Architecture-agnostic linear scaling of reasoning. *arXiv preprint arXiv:2510.06557*, 2025.

Anthropic. Claude Code. https://github.com/anthropics/claude-code, 2025. Agentic coding tool. Accessed: 2026-05-03.

Anthropic. Introducing Claude Opus 4.7. https://www.anthropic.com/news/claude-opus-4-7, April 2026. Published April 16, 2026. Accessed: 2026-05-05.

Ibragim Badertdinov, Alexander Golubev, Maksim Nekrashevich, Anton Shevtsov, Simon Karasik, Andrei Andriushchenko, Maria Trofimova, Daria Litvintseva, and Boris Yangel. SWE-rebench: An automated pipeline for task collection and decontaminated evaluation of software engineering agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026. URL https://openreview.net/forum?id=nMpJoVmRy1.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2025.

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

Guoxin Chen, Zile Qiao, Xuanzhong Chen, Donglei Yu, Haotian Xu, Xin Zhao, Ruihua Song, Wenbiao Yin, Huifeng Yin, Liwen Zhang, Kuan Li, Minpeng Liao, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Iterresearch: Rethinking long-horizon agents via markovian state reconstruction. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=qQ5MZ5Mx7p.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Sahel Sharifymoghaddam, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. Browsecomp-plus: A more fair and transparent evaluation benchmark of deep-research agent. *arXiv preprint arXiv:2508.06600*, 2025.

Yun Cheng, Xingyu Zhu, Haoyu Zhao, and Sanjeev Arora. Contextual drag: How errors in the context affect llm reasoning, 2026. URL https://arxiv.org/abs/2602.04288.

Cursor Research, :, Aaron Chan, Ahmed Shalaby, Alexander Wettig, Aman Sanger, Andrew Zhai, Anurag Ajay, Ashvin Nair, Charlie Snell, Chen Lu, Chen Shen, Emily Jia, Federico Cassano, Hanpeng Liu, Haoyu Chen, Henry Wildermuth, Jacob Jackson, Janet Li, Jediah Katz, Jiajun Yao, Joey Hejna, Josh Warner, Julius Vering, Kevin Frans, Lee Danilek, Less Wright, Lujing Cen, Luke Melas-Kyriazi, Michael Truell, Michiel de Jong, Naman Jain, Nate Schmidt, Nathan Wang, Niklas Muennighoff, Oleg Rybkin, Paul Loh, Phillip Kravtsov, Rishabh Yadav, Sahil Shah, Sam Kottler, Alexander M Rush, Shengtong Zhang, Shomil Jain, Sriram Sankar, Stefan Heule, Stuart H. Sul, Sualeh Asif, Victor Rong, Wanqi Zhu, William Lin, Yuchen Wu, Yuri Volkov, Yury Zemlyanskiy, Zack Holbrook, and Zhiyuan Zhang. Composer 2 technical report, 2026. URL https://arxiv.org/abs/2603.24477.

Sabri Eyuboglu, Ryan Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Liu, Will Tennien, Atri Rudra, James Zou, Azalia Mirhoseini, et al. Cartridges: Lightweight and general-purpose long context representations via self-study. *arXiv preprint arXiv:2506.06266*, 2025.

GLM Team, Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, Kedong Wang, Lucen Zhong, Mingdao Liu, Rui Lu, Shulin Cao, Xiaohan Zhang, Xuancheng Huang, Yao Wei, Yean Cheng, Yifan An, Yilin Niu, Yuanhao Wen, Yushi Bai, Zhengxiao Du, Zihan Wang, Zilin Zhu, Bohan Zhang, Bosi Wen, Bowen Wu, Bowen Xu, Can Huang, Casey Zhao, Changpeng Cai, Chao Yu, Chen Li, Chendi Ge, Chenghua Huang, Chenhui Zhang, Chenxi Xu, Chenzheng Zhu, Chuang Li, Congfeng Yin, Daoyan Lin, Dayong Yang, Dazhi Jiang, Ding Ai, Erle Zhu, Fei Wang, Gengzheng Pan, Guo Wang, Hailong Sun, Haitao Li, Haiyang Li, Haiyi Hu, Hanyu Zhang, Hao Peng, Hao Tai, Haoke Zhang, Haoran Wang, Haoyu Yang, He Liu, He Zhao, Hongwei Liu, Hongxi Yan, Huan Liu, Huilong Chen, Ji Li, Jiajing Zhao, Jiamin Ren, Jian Jiao, Jiani Zhao, Jianyang Yan, Jiaqi Wang, Jiayi Gui, Jiayue Zhao, Jie Liu, Jijie Li, Jing Li, Jing Lu, Jingsen Wang, Jingwei Yuan, Jingxuan Li, Jingzhao Du, Jinhua Du, Jinxin Liu, Junkai Zhi, Junli Gao, Ke Wang, Lekang Yang, Liang Xu, Lin Fan, Lindong Wu, Lintao Ding, Lu Wang, Man Zhang, Minghao Li, Minghuan Xu, Mingming Zhao, Mingshu Zhai, Pengfan Du, Qian Dong, Shangde Lei, Shangqing Tu, Shangtong Yang, Shaoyou Lu, Shijie Li, Shuang Li, Shuang-Li, Shuxun Yang, Sibo Yi, Tianshu Yu, Wei Tian, Weihan Wang, Wenbo Yu, Weng Lam Tam, Wenjie Liang, Wentao Liu, Xiao Wang, Xiaohan Jia, Xiaotao Gu, Xiaoying Ling, Xin Wang, Xing Fan, Xingru Pan, Xinyuan Zhang, Xinze Zhang, Xiuqing Fu, Xunkai Zhang, Yabo Xu, Yandong Wu, Yida Lu, Yidong Wang, Yilin Zhou, Yiming Pan, Ying Zhang, Yingli Wang, Yingru Li, Yinpei Su, Yipeng Geng, Yitong Zhu, Yongkun Yang, Yuhang Li, Yuhao Wu, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yuxuan Zhang, Zezhen Liu, Zhen Yang, Zhengda Zhou, Zhongpei Qiao, Zhuoer Feng, Zhuorui Liu, Zichen Zhang, Zihan Wang, Zijun Yao, Zikang Wang, Ziqiang Liu, Ziwei Chai, Zixuan Li, Zuodong Zhao, Wenguang Chen, Jidong Zhai, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models, 2025. URL https://arxiv.org/abs/2508.06471.

Google. Gemini 3.1 Pro. https://deepmind.google/models/gemini/pro/, 2026. Accessed: 2026-05-05.

Nikita Gupta, Riju Chatterjee, Lukas Haas, Connie Tao, Andrew Wang, Chang Liu, Hidekazu Oiwa, Elena Gribovskaya, Jan Ackermann, John Blitzer, Sasha Goldshtein, and Dipanjan Das. Deepsearchqa: Bridging the comprehensiveness gap for deep research agents, 2026. URL https://arxiv.org/abs/2601.20975.

Kelly Hong, Anton Troynikov, and Jeff Huber. Context rot: How increasing input tokens impacts llm performance. Technical report, Chroma, July 2025. URL https://trychroma.com/research/context-rot.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, S. H. Cai, Yuan Cao, Y. Charles, H. S. Che, Cheng Chen, Guanduo Chen, Huarong Chen, Jia Chen, Jiahao Chen, Jianlong Chen, Jun Chen, Kefan Chen, Liang Chen, Ruijue Chen, Xinhao Chen, Yanru Chen, Yanxu Chen, Yicun Chen, Yimin Chen, Yingjiang Chen, Yuankun Chen, Yujie Chen, Yutian Chen, Zhirong Chen, Ziwei Chen, Dazhi Cheng, Minghan Chu, Jialei Cui, Jiaqi Deng, Muxi Diao, Hao Ding, Mengfan Dong, Mengnan Dong, Yuxin Dong, Yuhao Dong, Angang Du, Chenzhuang Du, Dikang Du, Lingxiao Du, Yulun Du, Yu Fan, Shengjun Fang, Qiulin Feng, Yichen Feng, Garimugai Fu, Kelin Fu, Hongcheng Gao, Tong Gao, Yuyao Ge, Shangyi Geng, Chengyang Gong, Xiaochen Gong, Zhuoma Gongque, Qizheng Gu, Xinran Gu, Yicheng Gu, Longyu Guan, Yuanying Guo, Xiaoru Hao, Weiran He, Wenyang He, Yunjia He, Chao Hong, Hao Hu, Jiaxi Hu, Yangyang Hu, Zhenxing Hu, Ke Huang, Ruiyuan Huang, Weixiao Huang, Zhiqi Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yu Jing, Guokun Lai, Aidi Li, C. Li, Cheng Li, Fang Li, Guanghe Li, Guanyu Li, Haitao Li, Haoyang Li, Jia Li, Jingwei Li, Junxiong Li, Lincan Li, Mo Li, Weihong Li, Wentao Li, Xinhang Li, Xinhao Li, Yang Li, Yanhao Li, Yiwei Li, Yuxiao Li, Zhaowei Li, Zheming Li, Weilong Liao, Jiawei Lin, Xiaohan Lin, Zhishan Lin, Zichao Lin, Cheng Liu, Chenyu Liu, Hongzhang Liu, Liang Liu, Shaowei Liu, Shudong Liu, Shuran Liu, Tianwei Liu, Tianyu Liu, Weizhou Liu, Xiangyan Liu, Yangyang Liu, Yanming Liu, Yibo Liu, Yuanxin Liu, Yue Liu, Zhengying Liu, Zhongnuo Liu, Enzhe Lu, Haoyu Lu, Zhiyuan Lu, Junyu Luo, Tongxu Luo, Yashuo Luo, Long Ma, Yingwei Ma, Shaoguang Mao, Yuan Mei, Xin Men, Fanqing Meng, Zhiyong Meng, Yibo Miao, Minqing Ni, Kun Ouyang, Siyuan Pan, Bo Pang, Yuchao Qian, Ruoyu Qin, Zeyu Qin, Jiezhong Qiu, Bowen Qu, Zeyu Shang, Youbo Shao, Tianxiao Shen, Zhennan Shen, Juanfeng Shi, Lidong Shi, Shengyuan Shi, Feifan Song, Pengwei Song, Tianhui Song, Xiaoxi Song, Hongjin Su, Jianlin Su, Zhaochen Su, Lin Sui, Jinsong Sun, Junyao Sun, Tongyu Sun, Flood Sung, Yunpeng Tai, Chuning Tang, Heyi Tang, Xiaojuan Tang, Zhengyang Tang, Jiawen Tao, Shiyuan Teng, Chaoran Tian, Pengfei Tian, Ao Wang, Bowen Wang, Chensi Wang, Chuang Wang, Congcong Wang, Dingkun Wang, Dinglu Wang, Dongliang Wang, Feng Wang, Hailong Wang, Haiming Wang, Hengzhi Wang, Huaqing Wang, Hui Wang, Jiahao Wang, Jinhong Wang, Jiuzheng Wang, Kaixin Wang, Linian Wang, Qibin Wang, Shengjie Wang, Shuyi Wang, Si Wang, Wei Wang, Xiaochen Wang, Xinyuan Wang, Yao Wang, Yejie Wang, Yipu Wang, Yiqin Wang, Yucheng Wang, Yuzhi Wang, Zhaoji Wang, Zhaowei Wang, Zhengtao Wang, Zhexu Wang, Zihan Wang, Zizhe Wang, Chu Wei, Ming Wei, Chuan Wen, Zichen Wen, Chengjie Wu, Haoning Wu, Junyan Wu, Rucong Wu, Wenhao Wu, Yuefeng Wu, Yuhao Wu, Yuxin Wu, Zijian Wu, Chenjun Xiao, Jin Xie, Xiaotong Xie, Yuchong Xie, Yifei Xin, Bowei Xing, Boyu Xu, Jianfan Xu, Jing Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinbo Xu, Xinran Xu, Yangchuan Xu, Yichang Xu, Yuemeng Xu, Zelai Xu, Ziyao Xu, Junjie Yan, Yuzi Yan, Guangyao Yang, Hao Yang, Junwei Yang, Kai Yang, Ningyuan Yang, Ruihan Yang, Xiaofei Yang, Xinlong Yang, Ying Yang, Yi Yang, Yi Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Dan Ye, Wenjie Ye, Zhuorui Ye, Bohong Yin, Chengzhen Yu, Longhui Yu, Tao Yu, Tianxiang Yu, Enming Yuan, Mengjie Yuan, Xiaokun Yuan, Yang Yue, Weihao Zeng, Dunyuan Zha, Haobing Zhan, Dehao Zhang, Hao Zhang, Jin Zhang, Puqi Zhang, Qiao Zhang, Rui Zhang, Xiaobin Zhang, Y. Zhang, Yadong Zhang, Yangkun Zhang, Yichi Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yushun Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Chenguang Zhao, Feifan Zhao, Jinxiang Zhao, Shuai Zhao, Xiangyu Zhao, Yikai Zhao, Zijia Zhao, Huabin Zheng, Ruihan Zheng, Shaojie Zheng, Tengyang Zheng, Junfeng Zhong, Longguang Zhong, Weiming

Zhong, M. Zhou, Runjie Zhou, Xinyu Zhou, Zaida Zhou, Jinguo Zhu, Liya Zhu, Xinhao Zhu, Yuxuan Zhu, Zhen Zhu, Jingze Zhuang, Weiyu Zhuang, Ying Zou, and Xinxing Zu. Kimi k2.5: Visual agentic intelligence, 2026. URL https://arxiv.org/abs/2602.02276.

Vasilis Kontonis, Yuchen Zeng, Shivam Garg, Lingjiao Chen, Hao Tang, Ziyan Wang, Ahmed Awadallah, Eric Horvitz, John Langford, and Dimitris Papailiopoulos. Memento: Teaching llms to manage their own context, 2026. URL https://github.com/microsoft/memento.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. LLMs get lost in multi-turn conversation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=VKGTGGcwl6.

Yoonsang Lee, Howard Yen, Xi Ye, and Danqi Chen. Agentic aggregation for parallel scaling of long-horizon agentic tasks, 2026. URL https://arxiv.org/abs/2604.11753.

Michael Y. Li, Jubayer Ibn Hamid, Emily B. Fox, and Noah D. Goodman. Neural garbage collection: Learning to forget while learning to reason, 2026. URL https://arxiv.org/abs/2604.18002.

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. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

Jack Lindsey. Emergent Introspective Awareness in Large Language Models. *Transformer Circuits Thread*, October 2025. Published October 29, 2025. Accessed: 2026-05-05.

Junteng Liu, Yunji Li, Chi Zhang, Jingyang Li, Aili Chen, Ke Ji, Weiyu Cheng, Zijia Wu, Chengyu Du, Qidi Xu, Jiayuan Song, Zhengmao Zhu, Wenhu Chen, Pengyu Zhao, and Junxian He. Webexplorer: Explore and evolve for training long-horizon web agents, 2025. URL https://arxiv.org/abs/2509.06501.

Thang Luong, Dawsen Hwang, Hoang H. Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, Alex Zhai, Clara Huiyi Hu, Henryk Michalewski, Jimin Kim, Jeonghyun Ahn, Junhwi Bae, Xingyou Song, Trieu H. Trinh, Quoc V. Le, and Junehyuk Jung. Towards robust mathematical reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025. URL https://aclanthology.org/2025.emnlp-main.1794/.

Lingrui Mei, Jiayu Yao, Yuyao Ge, Yiwei Wang, Baolong Bi, Yujun Cai, Jiazhi Liu, Mingyu Li, Zhong-Zhi Li, Duzhen Zhang, et al. A survey of context engineering for large language models. *arXiv preprint arXiv:2507.13334*, 2025.

METR. Time horizon 1.1. https://metr.org/blog/2026-1-29-time-horizon-1-1/, 01 2026.

MiMo Team, Bangjun Xiao, Bingquan Xia, Bo Yang, Bofei Gao, Bowen Shen, Chen Zhang, Chenhong He, Chiheng Lou, Fuli Luo, Gang Wang, Gang Xie, Hailin Zhang, Hanglong Lv, Hanyu Li, Heyu Chen, Hongshen Xu, Houbin Zhang, Huaqiu Liu, Jiangshan Duo, Jianyu Wei, Jiebao Xiao, Jinhao Dong, Jun Shi, Junhao Hu, Kainan Bao, Kang Zhou, Lei Li, Liang Zhao, Linghao Zhang, Peidian Li, Qianli Chen, Shaohui Liu, Shihua Yu, Shijie Cao, Shimao Chen, Shouqiu Yu, Shuo Liu, Tianling Zhou, Weijiang Su, Weikun Wang, Wenhan Ma, Xiangwei Deng, Bohan Mao, Bowen Ye, Can Cai, Chenghua Wang, Chengxuan Zhu, Chong Ma, Chun Chen, Chunan Li, Dawei Zhu, Deshan Xiao, Dong Zhang, Duo Zhang, Fangyue Liu, Feiyu Yang, Fengyuan Shi, Guoan Wang, Hao Tian, Hao Wu, Heng Qu, Hongfei Yi, Hongxu An, Hongyi Guan, Xing Zhang, Yifan Song, Yihan Yan, Yihao Zhao, Yingchun Lai, Yizhao Gao, Yu Cheng, Yuanyuan Tian, Yudong Wang, Zhen Tang, Zhengju Tang, Zhengtao Wen, Zhichao Song, Zhixian Zheng, Zihan Jiang, Jian Wen, Jiarui Sun, Jiawei Li, Jinlong Xue, Jun Xia, Kai Fang, Menghang Zhu, Nuo Chen, Qian Tu, Qihao Zhang, Qiying Wang, Rang Li, Rui Ma, Shaolei Zhang, Shengfan Wang, Shicheng Li, Shuhao Gu, Shuhuai Ren, Sirui Deng, Tao Guo, Tianyang Lu, Weiji Zhuang, Weikang Zhang, Weimin Xiong,

Wenshan Huang, Wenyu Yang, Xin Zhang, Xing Yong, Xu Wang, Xueyang Xie, Yilin Jiang, Yixin Yang, Yongzhe He, Yu Tu, Yuanliang Dong, Yuchen Liu, Yue Ma, Yue Yu, Yuxing Xiang, Zhaojun Huang, Zhenru Lin, Zhipeng Xu, Zhiyang Chen, Zhonghua Deng, Zihan Zhang, and Zihao Yue. Mimo-v2-flash technical report, 2026. URL https://arxiv.org/abs/2601.02780.

MiniMax Team. MiniMax M2.5: Built for real-world productivity. https://www.minimax.io/news/minimax-m25, February 2026. Accessed: 2026-04-28.

MiroMind Team, Song Bai, Lidong Bing, Carson Chen, Guanzheng Chen, Yuntao Chen, Zhe Chen, Ziyi Chen, Jifeng Dai, Xuan Dong, Wenhan Dou, Yue Deng, Yunjie Fu, Junqi Ge, Chenxia Han, Tammy Huang, Zhenhang Huang, Jerry Jiao, Shilei Jiang, Tianyu Jiao, Xiaoqi Jian, Lei Lei, Ruilin Li, Gen Luo, Tiantong Li, Xiang Lin, Ziyuan Liu, Zhiqi Li, Jie Ni, Qiang Ren, Pax Sun, Shiqian Su, Chenxin Tao, Bin Wang, Wenhai Wang, Haonan Wang, James Wang, Jin Wang, Jojo Wang, Letian Wang, Shizun Wang, Weizhi Wang, Zixuan Wang, Jinfan Xu, Sen Xing, Chenyu Yang, Hai Ye, Jiaheng Yu, Yue Yu, Muyan Zhong, Tianchen Zhao, Xizhou Zhu, Yanpeng Zhou, Yifan Zhang, and Zhi Zhu. Mirothinker: Pushing the performance boundaries of open-source research agents via model, context, and interactive scaling, 2026. URL https://arxiv.org/abs/2511.11793.

Purbesh Mitra and Sennur Ulukus. Motif: Modular thinking via reinforcement fine-tuning in llms. *arXiv preprint arXiv:2507.02851*, 2025.

Giovanni Monea, Yair Feldman, Shankar Padmanabhan, Kianté Brantley, and Yoav Artzi. Breadcrumbs reasoning: Memory-efficient reasoning with compression beacons. *arXiv preprint arXiv:2510.13797*, 2025.

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

OpenAI. GPT-5.5 System Card. https://deploymentsafety.openai.com/gpt-5-5, April 2026. Published April 23, 2026. Accessed: 2026-05-05.

Junyoung Park, Dalton Jones, Matthew J Morse, Raghavv Goel, Mingu Lee, and Chris Lott. Keydiff: Key similarity-based kv cache eviction for long-context llm inference in resource-constrained environments. *arXiv preprint arXiv:2504.15364*, 2025.

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

Jiaqi Shao, Yufeng Miao, Wei Zhang, and Bing Luo. Foldact: Efficient and stable context folding for long-horizon search agents, 2025. URL https://arxiv.org/abs/2512.22733.

Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. Scaling long-horizon llm agent via context-folding, 2025. URL https://arxiv.org/abs/2510.11967.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents, 2025. URL https://arxiv.org/abs/2504.12516.

Ian Wu, Yuxiao Qu, Amrith Setlur, and Aviral Kumar. Reasoning cache: Continual improvement over long horizons via short-horizon rl, 2026a. URL https://arxiv.org/abs/2602.03773.

Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang Zhang, Xinmiao Yu, Dingchu Zhang, Yong Jiang, Pengjun Xie, Fei Huang, Minhao Cheng, Shuai Wang, Hong Cheng, and Jingren Zhou. Resum: Unlocking long-horizon search intelligence via context summarization, 2026b. URL https://arxiv.org/abs/2509.13313.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

Yuchen Yan, Yongliang Shen, Yang Liu, Jin Jiang, Mengdi Zhang, Jian Shao, and Yueting Zhuang. Inftythink: Breaking the length limits of long-context reasoning in large language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=T1h5em349L.

Zhicheng Yang, Zhijiang Guo, Yinya Huang, Yongxin Wang, Wenlei Shi, Yiwei Wang, Xiaodan Liang, and Jing Tang. Accordion-thinking: Self-regulated step summaries for efficient and readable llm reasoning, 2026. URL https://arxiv.org/abs/2602.03249.

Rui Ye, Zhongwang Zhang, Kuan Li, Huifeng Yin, Zhengwei Tao, Yida Zhao, Liangcai Su, Liwen Zhang, Zile Qiao, Xinyu Wang, Pengjun Xie, Fei Huang, Siheng Chen, Jingren Zhou, and Yong Jiang. Agentfold: Long-horizon web agents with proactive context management, 2025. URL https://arxiv.org/abs/2510.24699.

Jialiang Zhu, Gongrui Zhang, Xiaolong Ma, Lin Xu, Miaosen Zhang, Ruiqi Yang, Song Wang, Kai Qiu, Zhirong Wu, Qi Dai, Ruichun Ma, Bei Liu, Yifan Yang, Chong Luo, Zhengyuan Yang, Linjie Li, Lijuan Wang, Weizhu Chen, Xin Geng, and Baining Guo. Re-trac: Recursive trajectory compression for deep search agents, 2026a. URL https://arxiv.org/abs/2602.02486.

Yuqi Zhu, Jintian Zhang, Zhenjie Wan, Yujie Luo, Shuofei Qiao, Zhengke Gui, Da Zheng, Lei Liang, Huajun Chen, and Ningyu Zhang. Lightthinker++: From reasoning compression to memory management, 2026b. URL https://arxiv.org/abs/2604.03679.

Chunsheng Zuo, Liaoyaqi Wang, William Jurayj, William Fleshman, and Benjamin Van Durme. Rethinking lora memory through the lens of kv cache compression, 2026. URL https://arxiv.org/abs/2606.05698.

Adam Zweiger, Xinghong Fu, Han Guo, and Yoon Kim. Fast kv compaction via attention matching. *arXiv preprint arXiv:2602.16284*, 2026.

A Math experimental setup

Models. We evaluate four open-weight Qwen models on competition math: two instruction-tuned models (`Qwen3-4B-Instruct-2507` and `Qwen3-30B-A3B-Instruct-2507`) and two thinking-disabled variants (`Qwen3.5-4B` and `Qwen3.5-9B`). All checkpoints are loaded from Hugging Face and served via vLLM [Kwon et al., 2023]. The 30B-A3B model uses tensor-parallel size 2; all others run on a single H200 GPU.

Benchmarks. IMO-Answerbench [Luong et al., 2025], HMMT November 2025, and HMMT February 2026. Final answers are extracted and scored with the `math_verify` package.

Sampling parameters. We use temperature 1.0 and top-p 0.7 for all models, generating $n = 16$ samples per question and reporting mean $\pm$ std over samples. Each generation round is capped at 16,384 tokens (`-num_tokens 16384`); the per-trajectory round budget is `-max_rounds 12`, and the summarizer output is hard-truncated at 512 tokens (`-max_summary_tokens 512`).

Scaffold. We adopt the ReasoningCache scaffold of Wu et al. [2026a]: at each round, the model is prompted to either solve from scratch or improve on a summary of the previous attempt. The continuation prompt issued to the model after every refinement is reproduced verbatim in Box A.

Methods. BASELINE runs the model with a single 16k-token budget (`-max_rounds 1`) and no summarization. FIXED-INTERVAL SUMMARY sets `-force_tool_calls`, which fires the summarizer at the end of every round (i.e. every 16,384 generated tokens) regardless of trajectory state. SELFCOMPACT exposes `<tool_call>` as a vLLM stop token so that the model autonomously emits the summarizer; the round ends and the summary fires whenever the stop token appears. SELF-COMPACT W/ RUBRICS interposes the rubric probe in Box A between rounds: if the rubric fires COMPRESS the summarizer is invoked, otherwise the model continues from the unmodified prefix.

Math rubric prompt (ANSWER / STUCK / HAS-NEXT; appended to $(x, y_{1:t})$ at every round boundary)

Judge your math-solving state from the conversation above. Answer Q1…Q3 with Y or N and one short evidence phrase. Answers without evidence must be N. "Non-trivial fact" = a new equation, reduction, bound, case-elimination, or counterexample, not a paraphrase.

Q1 ANSWER: The latest round states a specific final answer (a `\boxed{}` expression or "Final Answer: …"), not merely a partial result. *If Y*, quote the answer verbatim. *If N*, state what is still unknown.

Q2 STUCK: Your last 2 rounds added no non-trivial fact — only paraphrases or abandoned attempts. *If Y*, name the two rounds and write "no new fact". *If N*, name one non-trivial fact from the last 2 rounds.

Q3 HAS-NEXT: You can write the exact next step (case split, substitution, verification, lemma to prove). *If Y*, write the step as one imperative sentence. *If N*, write "NONE".

Output: exactly 3 lines, no preamble or trailing text.

```
Q1: Y/N -- <evidence>
Q2: Y/N -- <evidence>
Q3: Y/N -- <evidence>
```

Fire rule: COMPRESS iff Q1 = Y $\vee$ (Q2 = Y $\wedge$ Q3 = Y). Branch A (Q1 = Y) fires a lock-in re-prompt that preserves the boxed answer; Branch B (Q2 = Y $\wedge$ Q3 = Y) fires a reset that preserves the named next step.

Math summarizer prompt (preserve-answer v3; replaces $y_{1:t}$ with $\tilde{y}$ when fired)

Context Refinement Prompt:

Original Prompt: `{original_prompt}`
Partial Generation: `{partial_generation}`

Your task is to create a compressed summary for another model to continue solving from.

RULES:

1. If a final answer (e.g., `\boxed{}`) was found, PRESERVE IT at the end of your summary.
2. Keep key insights, important calculations, and the reasoning path.
3. Remove redundant text, false starts, and unnecessary repetition.
4. If the answer seems wrong or unverified, note that verification is needed.
5. Be concise but preserve all critical mathematical steps.

Output format:

- Key insights and progress made.
- Important intermediate results.
- If found: "Final Answer: [the answer]" or the `\boxed{}` expression.
- If not solved: what still needs to be done.

Continuation prompt (issued to π at the start of every round, with $\tilde{y}$ filled in if available)

You are given a maths problem. You may also be given a summary of a previous attempt to solve it. This previous attempt may or may not be correct.

PROBLEM `{problem}`
SUMMARY OF PREVIOUS ATTEMPT `{summary}`

INSTRUCTIONS
If no summary of a previous attempt is provided, solve the problem from scratch.
If a summary of a previous attempt is provided, your task is to improve upon this attempt. You should rely on this summary to guide your thinking. Some strategies you could use include:

- Verifying the previous solution.
- Proving the result in a different way.
- Finding alternative problem-solving strategies.
- Continuing from where the previous solution left off, assuming that the previous solution is incomplete.

Reason step-by-step and return your final answer in `\boxed{}`.

B Agentic-search experimental setup

Models and serving. The three deployed agents (`GLM-4.7-Flash`, `MiniMax-M2.5`, `Mimo-V2-Flash`) are accessed through OpenRouter via the AggAgent runtime [Lee et al., 2026]. We use the unmodified `react_agent_selfcheck` scaffold; the rubric and summarizer prompts are reproduced verbatim in Box B and Box B.

Benchmarks. BrowseComp [Wei et al., 2025], BrowseComp-Plus [Chen et al., 2025], and DeepSearchQA [Gupta et al., 2026]. Following Sun et al. [2025] and Lee et al. [2026], we sub-sample 150 questions per benchmark.

Sampling parameters. All three models share the same decoding configuration: temperature 1.0, top-p 0.95, max output tokens per call 10,000, parallel-tool-calls disabled, and a per-trajectory cap of 100 LLM calls. Per-model context windows, the 30%-of-context trigger used by FIXED-INTERVAL SUMMARY and the rubric's token-pct backstop, and model-specific decoding flags are listed in Table 7.

Table 7: Per-model context windows, 30%-of-context summarization trigger, and decoding flags for the three OpenRouter agents. GLM-4.7-Flash keeps the thinking trace visible to the rubric probe via `enable_thinking=True` and `clear_thinking=False`.

Model	Ctx window	30% trigger	Model-specific flags
`GLM-4.7-Flash`	128,000	38,400	`enable_thinking=True`, `clear_thinking=False`
`MiniMax-M2.5`	196,608	58,982	`reasoning_split=True`
`Mimo-V2-Flash`	262,144	78,643	(default OpenRouter body)

Scaffold details. A single trajectory is the standard ReAct loop: the agent emits a tool call (`search` and either `visit` for BrowseComp/DeepSearchQA or `get_document_bcp` for BrowseComp-Plus), receives the result, reasons, and either issues another tool call or commits a final answer. For BrowseComp and DeepSearchQA, `search` is backed by the Serper API (`https://serper.dev/`) and `visit` fetches and extracts page content via crawl4ai (`https://github.com/unclecode/crawl4ai`); for BrowseComp-Plus, `search` and `get_document_bcp` resolve against the benchmark's released corpus. The rubric in Box B is appended to a *copy* of the trajectory at each round boundary so the probe does not pollute the rolling cache. COMPRESS fires only when four gates pass simultaneously: (ROUND) iteration ≥ 3, (TOKENS) running prompt length $\geq 40{,}000$, (CAP) total summaries so far < 1, and (PERIOD) ≥ 2 rounds elapsed since the last probe. The token-pct backstop forces a COMPRESS once the prompt crosses $0.30 \cdot$ ctx_window regardless of the rubric.

Search rubric probe (appended to $(x, y_{1:t})$ at every round boundary)

You are about to decide whether to compress your conversation history into a summary that REPLACES the full history above. After compression, you continue research from only `[system, original_question, assistant_summary, user_continue]`. Compression is irreversible: anything not preserved in the summary is gone.

Compression is safe ONLY when ALL FOUR of the following hold:

(C1) the trajectory has reached a closed unit (not mid-thought),
(C2) the essential information is reducible to 3–5 cite-able facts without loss,
(C3) something has progressed since the last compression,
(N1) you are NOT currently stuck in a way summarization would mask.

Answer C1, C2, C3, N1 honestly. Each Y answer requires verbatim evidence quoted from the trajectory above; answers without evidence default to N.

C1 CLOSED-UNIT: The most recent assistant message is a closed unit — a completed tool call whose result is now visible, or a completed sub-analysis with a clear stopping point. It is NOT mid-sentence reasoning ("Let me now check…", "I should next look at…"), and not a half-formulated query. *If Y*, quote the closing fragment of the last assistant message. *If N*, quote the open fragment that shows the trajectory is mid-thought.

C2 SUMMARIZABLE: You can write 3–5 essential facts (with verbatim citations from the trajectory) that future-you needs to continue research after compression. Each fact must be a single concrete statement: a name, date, URL, quoted claim, or resolved sub-question. Answer N if the trajectory's value is dispersed across many small inferences (e.g., a list of dead-end queries needed to avoid retries, negative results that constrain hypothesis space) that would be lost without the dispersal. *If Y*, list the 3–5 facts numbered, each with a verbatim citation in quotes, separated by " | ". *If N*, name in one sentence the class of information that would be lost.

C3 PROGRESS: Since the most recent compression (or since the start of the conversation if none), you have either obtained a new concrete fact (name, date, URL, or quoted claim) OR refined the sub-question being pursued. *If Y*, name the new fact or refined sub-question. *If N*, state that you are returning the same state you compressed from.

N1 STUCK: At least 3 of your last 4 search queries returned no new URL or fact (i.e., were duplicates or returned already-known content). If you have made fewer than 4 searches total, answer N. *If Y*, name 1 distinct strategy you have NOT yet tried (different tool, different query type, different angle on the question). *If N*, name one new URL or fact obtained recently.

Output: exactly 4 lines, no preamble or trailing text.

```
C1: Y/N -- <evidence>
C2: Y/N -- <if Y: 1. fact "citation" | 2. fact "citation" |
            3. fact "citation"; if N: <class of info lost>>
C3: Y/N -- <evidence>
N1: Y/N -- <evidence>
```

Fire rule: COMPRESS iff C1 = Y ∧ C2 = Y ∧ C3 = Y ∧ N1 = N; otherwise CONTINUE.

Summarizer. The search summarizer is the `webresummer` variant of Wu et al. [2026b], reproduced verbatim in Box B below; we use it unmodified for all three deployed agents.

Summarizer prompt ($\mathcal{S}$ instruction appended to $(x, y_{1:t})$)

You are an expert at analyzing conversation history and extracting relevant information. Your task is to thoroughly evaluate the conversation history above and the user's original question to provide a summary that will REPLACE the full conversation history when you continue.

Task Guidelines
1. Information Analysis:

- Carefully analyze the conversation history to identify truly useful information.
- Focus on information that directly contributes to answering the question.
- Do NOT make assumptions, guesses, or inferences beyond what is explicitly stated in the conversation.
- If information is missing or unclear, do NOT include it in your summary.

2. Summary Requirements:

- Extract only the most relevant information that is explicitly present in the conversation.
- Synthesize information from multiple exchanges when relevant.
- Only include information that is certain and clearly stated in the conversation.
- Do NOT output or mention any information that is uncertain, insufficient, or cannot be confirmed from the conversation.

C Cost analysis of summarization

Cost model (single-rate). In the main text, for all OpenRouter-hosted agents we charge each question with the additive token-cost

$$\textsc{Cost}(q) = \frac{1}{10^6}\Big(p_{\text{cache}} \cdot N_{\text{prompt}} + p_{\text{out}} \cdot N_{\text{out}}\Big),$$

where N_{prompt} is the cumulative prompt tokens summed across *every* LLM call in the trajectory assistant turns, the rubric probe, and the summarizer call, and N_{out} is the cumulative completion tokens over the same set of calls. We charge every prompt token at the cache rate p_{cache} rather than separately tracking which tokens were a cache hit and which were a first-time prefill. This is a conservative single-rate approximation: the first-time prefill tokens — the question, each assistant turn, and each tool observation, counted once when they first enter the prefix — are only a small fraction (empirically 5–8%) of N_{prompt}, which counts each token once per call it survives, so the effective per-token rate sits close to p_{cache} even when provider-side caching misses occasionally. Appendix D relaxes this approximation, splitting N_{prompt} into first-time prefill N_{prefill} (billed at the uncached rate p_{in}) and cache reads N_{cached} (billed at p_{cache}) and reporting the resulting two-rate cost in Table 9. Per-1M token prices are listed in Table 8.

Calculation procedure. For each of the 150 sampled questions per benchmark, we (i) load the trajectory's per-call usage records logged by the AggAgent runtime, (ii) accumulate $(N_{\text{prompt}}, N_{\text{out}})$ as defined above, (iii) apply the model's prices from Table 8, and (iv) report the mean across the 150 questions in Table 4 (units: USD per question).

Table 8: OpenRouter prices in USD per 1,000,000 tokens.

Model	p_{in}	p_{cache}	p_{out}
`GLM-4.7-Flash`	0.07	0.01	0.40
`MiMo-V2-Flash`	0.10	0.01	0.30
`MiniMax-M2.5`	0.30	0.03	1.20

Where compaction saves cost. Under the single-rate cache billing above, the savings from compaction come entirely from shortening every *subsequent* call's prompt. After a fire that collapses a T-token prefix to a $\tilde{t}$-token summary, every later call pays $p_{\text{cache}} \cdot \tilde{t}$ instead of $p_{\text{cache}} \cdot T$, a $T/\tilde{t}$ ratio per call. Empirically summarization collapses a 50–100k trajectory into $\tilde{t} \approx$ 1–3k, giving 20–80$\times$ shrinkage of the post-compact prompt. The rubric probe and summarizer both append to the live prefix so the running prompt is preserved during the probe itself; the probe contributes only its own $\sim$60-token verdict to N_{out}, and the summarizer contributes $\tilde{t}$ tokens once, both negligible relative to the future-call savings amortized across the rest of the trajectory.

Math. The Qwen runs are served locally with vLLM, so cost collapses to a wall-clock trade-off rather than a dollar one. We report the per-question budget tag `[Xk]` in Table 1: the average sum of generated tokens (`rounds` $\times$ `num_tokens`) plus refinement tokens across the $n = 16$ samples. FIXED-INTERVAL SUMMARY's budget is matched to SELFCOMPACT within $\pm$3k tokens so all four rows in each model block compete at identical compute.

D Trajectory token consumption

Token lifecycle. A trajectory under SELFCOMPACT looks like $Q \to A_1 \to A_2 \to \cdots \to A_k \to S \to A'_1 \to A'_2 \to \cdots$, where Q is the original question (plus system prompt), each A_t is one assistant turn (model output for tool call or final answer), and S is a summary the model emits when the rubric fires COMPRESS. Each token a model produces is billed three different ways over its lifetime:

- *Output* — when the model first emits the tokens (one charge).
- *Prefill* (uncached input) — on the immediately next LLM call, when the tokens are appended to the prefix and seen by the provider for the first time (one charge).
- *Cached input* — on every subsequent LLM call until the prefix is reset (one charge per reuse).

Concretely, a single A_t at round t in a trajectory of R rounds contributes once to *Output*, once to *Prefill*, and $R - t$ times to *Cached*. A summary S emitted at compaction is billed once as output, once as prefill on the first post-compact call, then as cached input until the next compaction (or end of trajectory).

Per-question token totals. Table 9 reports, for each (model, benchmark, method) cell, the cumulative prompt tokens N_{prompt} and completion tokens N_{out} summed across all LLM calls (assistant turns, rubric probes, summarizer calls), together with the split of N_{prompt} into first-time prefill N_{prefill} and cache reads N_{cached}. The single-rate cost reported in the main text (Table 4) recovers directly from the N_{prompt} column as $(p_{\text{cache}}\, N_{\text{prompt}} + p_{\text{out}}\, N_{\text{out}})/10^6$. The **Cost** column of Table 9 instead applies the finer two-rate billing, $(p_{\text{in}}\, N_{\text{prefill}} + p_{\text{cache}}\, N_{\text{cached}} + p_{\text{out}}\, N_{\text{out}})/10^6$, charging the first-time prefill at the higher uncached rate p_{in}; this raises each entry modestly above the single-rate value in Table 4 but preserves the headline comparison that SELFCOMPACT costs less than the no-compaction baseline.

Table 9: Per-question token accounting across all three benchmarks, splitting the cumulative prompt N_{prompt} into first-time prefill (N_{prefill}) and cache reads (N_{cached}), alongside output (N_{out}). Cost uses the two-rate billing $N_{\text{prefill}}\, p_{\text{in}} + N_{\text{cached}}\, p_{\text{cache}} + N_{\text{out}}\, p_{\text{out}}$ with the prices in Table 8; this is finer than the single-rate approximation behind Table 4 and so gives slightly higher costs.

Model	**Benchmark**	**Method**	N_{prompt}	N_{prefill}	N_{cached}	N_{out}	**Cost**
`GLM-4.7-Flash`	BrowseComp-Plus	No Compaction	11.6M	0.58M	11.0M	8.9k	\$0.15
		Fixed-interval	3.6M	0.29M	3.31M	9k	\$0.06
		SELFCOMPACT	3.6M	0.29M	3.31M	9k	\$0.06
	BrowseComp	No Compaction	13.6M	0.68M	12.9M	8.9k	\$0.18
		Fixed-interval	5.6M	0.45M	5.15M	9k	\$0.09
		SELFCOMPACT	9.6M	0.77M	8.83M	9k	\$0.15
	DeepSearchQA	No Compaction	12.6M	0.63M	12.0M	8.9k	\$0.17
		Fixed-interval	4.6M	0.37M	4.23M	9k	\$0.07
		SELFCOMPACT	6.6M	0.53M	6.07M	9k	\$0.10
`MiniMax-M2.5`	BrowseComp-Plus	No Compaction	5.7M	0.29M	5.41M	16.6k	\$0.27
		Fixed-interval	0.7M	0.06M	0.64M	16.9k	\$0.06
		SELFCOMPACT	1.7M	0.14M	1.56M	17.4k	\$0.11
	BrowseComp	No Compaction	5.7M	0.29M	5.41M	16.6k	\$0.27
		Fixed-interval	1.7M	0.13M	1.57M	16.9k	\$0.11
		SELFCOMPACT	2.3M	0.18M	2.12M	17.4k	\$0.14
	DeepSearchQA	No Compaction	5.7M	0.29M	5.41M	16.6k	\$0.27
		Fixed-interval	1.3M	0.11M	1.19M	16.9k	\$0.09
		SELFCOMPACT	2.0M	0.16M	1.84M	17.4k	\$0.12
`MiMo-V2-Flash`	BrowseComp-Plus	No Compaction	23.8M	1.19M	22.6M	6.8k	\$0.35
		Fixed-interval	13.8M	1.10M	12.7M	7.0k	\$0.24
		SELFCOMPACT	15.8M	1.26M	14.5M	7.2k	\$0.27
	BrowseComp	No Compaction	26.8M	1.34M	25.5M	6.8k	\$0.39
		Fixed-interval	13.8M	1.10M	12.7M	7.0k	\$0.24
		SELFCOMPACT	10.8M	0.86M	9.94M	7.2k	\$0.19
	DeepSearchQA	No Compaction	24.8M	1.24M	23.6M	6.8k	\$0.36
		Fixed-interval	13.8M	1.10M	12.7M	7.0k	\$0.24
		SELFCOMPACT	12.8M	1.02M	11.8M	7.2k	\$0.22

Table 10: Per-question token consumption for the math benchmarks (IMO-Answerbench / HMMT Nov 25 / HMMT Feb 26 averaged). Prompt is the cumulative refinement summaries fed back into the next round; output is model generation per round. The Output column matches the `[Xk]` tag in Table 1; Total is Prompt + Output.

Method	**Prompt (cum.)**	**Output**	**Total**
`Qwen3-4B-Instruct-2507`			
No Compaction	1.2k	16.0k	17k
Fixed-Interval	7.1k	44.1k	51k
SELFCOMPACT	7.9k	47.9k	56k
`Qwen3-30B-A3B-Instruct-2507`			
No Compaction	1.2k	16.0k	17k
Fixed-Interval	4.1k	25.6k	30k
SELFCOMPACT	4.6k	28.6k	33k
`Qwen3.5-9B (Thinking Disabled)`			
No Compaction	1.2k	16.0k	17k
Fixed-Interval	14.4k	89.0k	103k
SELFCOMPACT	15.0k	92.5k	107k
`Qwen3.5-4B (Thinking Disabled)`			
No Compaction	1.2k	16.0k	17k
Fixed-Interval	10.1k	63.7k	74k
SELFCOMPACT	10.7k	66.6k	78k

E Qualitative trajectories

We walk through three BrowseComp-Plus questions on `MiniMax-M2.5`, each run two ways: **fixed-interval** compaction (summarize whenever the prompt passes 30% of the context window) and **SELFCOMPACT** (summarize only when the rubric judges that a sub-task has closed). On all three questions both fixed-interval *and* the no-compaction baseline answer incorrectly, while SELFCOMPACT answers correctly — so the difference is *when* compaction fires, not whether it fires.

How to read these traces. Each turn is one `search` or `get_document` call. Fixed-interval compacts on a blind clock: every time the running prompt refills to ≈60k tokens (30% of MiniMax-M2.5's window) it summarizes whatever is currently in context — including dead-end searches. SELFCOMPACT probes a rubric at intervals and compacts only when a sub-task has genuinely resolved, so it can let the prompt grow well past 30% while the agent is still exploring. We write each compaction as (prefix tokens → summary tokens).

Case A: Whitesnake — fixed-interval stays anchored, SELFCOMPACT breaks out

Question. Name the band formed between 1970 and 1990 by a musician who played in a rock band that sold 100M+ records, began performing as a teenager, was married three times, has one daughter and one son, attended art college, and worked in a boutique. **Gold:** *Whitesnake* (the musician is David Coverdale, ex-Deep Purple).
Fixed-interval @ 30% (wrong: "Paul McCartney formed Wings").

- **Turns 1–20.** Cycles famous British rockers against the clues — Keith Richards, Pete Townshend, Malcolm McLaren, Bryan Ferry — none of whom fit.
- **Compactions at turns 21, 41, 61, 82** (each ≈60k → 1–3k). Each summary re-dumps the same ruled-out-names search log, so the agent resumes from the identical dead-end shortlist.
- **Turns 22–100.** Keeps re-trying the same pool (Brian May, Mick Jagger, Ronnie Wood, Paul McCartney), never tests David Coverdale, hits the 100-search cap, and falls back to a generic guess.

SELFCOMPACT (correct: *Whitesnake*).

- **Turns 1–40.** The same broad cycling (Freddie Mercury, Bowie, Boy George, Stevie Nicks); the rubric keeps continuing — no sub-task has closed — and lets the prompt grow to ≈118k.
- **One compaction at turn 41** (118.6k → 1.1k), distilling the constraints rather than the ruled-out shortlist.
- **Turns 46–57.** No longer re-primed with the old names, the agent tries a fresh candidate — David Coverdale — confirms Deep Purple (100M+ records), art college, and the boutique job, and answers *Whitesnake*.

Fixed-interval compacted four times but kept re-seeding the same wrong shortlist; SELFCOMPACT's single late compaction cleared it and let the agent escape to the right musician.

Case B: Majida El Roumi — fixed-interval freezes a wrong lead

Question. An artist released their first solo album between 1975 and 1978, served as a UN Goodwill Ambassador, and made a song (released 2004–2007) that is a rendition of a 1956–1959 classical piece; a different artist — who started singing at age eight and cited Barbra Streisand and Queen — released a version between 1994 and 1997. Name the first artist. **Gold:** *Majida El Roumi* (her "Habibi" is built on Albinoni's *Adagio*).
Fixed-interval @ 30% (wrong: "None").

- **Turns 1–20.** Chases both threads but latches onto the wrong classical piece — Rachmaninoff, via Céline Dion's "All by Myself".
- **Compactions at turns 21, 41, 61, 81** (≈60k each). Every summary carries the "All by Myself" lead forward, so the agent stays on Western pop (Nana Mouskouri, Eric Carmen, Cyndi Lauper, Angélique Kidjo).
- **Turns 22–100.** Never reconsiders the piece or tests an Arabic artist; hits the cap and returns "None".

SELFCOMPACT (correct: *Majida El Roumi*).

- **Turns 1–23.** Searches both threads and at turns 18–20 corrects the piece — Albinoni's *Adagio*, not Rachmaninoff — then looks for an Adagio cover by a UN-Ambassador artist.
- **One compaction at turn 24** (70.2k → 0.9k), consolidating the corrected lead.
- **Turns 25–42.** Zeroes in on Majida El Roumi and her song "Habibi", confirms the album dates, and answers correctly.

The fixed clock froze an early wrong guess into every summary; SELFCOMPACT compacted only after the agent had corrected the lead to Albinoni's *Adagio*.

Case C: Raheem Sterling — fixed-interval never reaches the match

Question. A league football match (1992–2023) in a European capital, attendance 61,700–61,906; Team Y scored in the 6th minute (a left-footed player), Team X scored in the 35th and 75th minutes and in stoppage time (90'+4', 90'+7'). Name the player who assisted the third goal (the 75th-minute one). **Gold:** *Raheem Sterling* (Tottenham 1–4 Chelsea, 6 Nov 2023).
Fixed-interval @ 30% (wrong: "Unable to determine").

- **Turns 1–22.** Searches the goal-minute fingerprint (6'/35'/75'/90+4'/90+7') and the attendance band; the literal-minute queries return nothing relevant. Guesses Sevilla–Roma.
- **Compactions at turns 23, 45, 67, 88** (≈60k each). Each summary re-dumps the same fruitless minute-by-minute log, and the agent keeps trying stadiums and teams at random (Stadio Olimpico, PSG, Man Utd, Moscow).
- **Turns 24–100.** Never tests Tottenham–Chelsea; hits the cap and gives up.

SELFCOMPACT (correct: *Raheem Sterling*).

- **Turns 1–22.** The same fingerprint searches, also unproductive.
- **Turns 23–25.** Tries the right pairing — Tottenham vs Chelsea, 4–1, November 2023 — and hits the match.
- **One compaction at turn 28** (70k → 0.8k), consolidating the identified match; turns 29–41 confirm the attendance, Kulusevski's left-footed 6th-minute goal, and Sterling's assist on the 75th-minute goal.

Both arms flailed on the goal-minute fingerprint, but SELFCOMPACT kept exploring until the match was found and compacted only then; the fixed clock kept interrupting to re-summarize dead ends and never reached the pairing.