The Complexity Trap: Simple Observation Masking Is as Efficient as LLM Summarization for Agent Context Management

Tobias Lindenbauer[1,2] **Igor Slinko**[1] **Ludwig Felder**[2]
Egor Bogomolov[1] **Yaroslav Zharov**[1]
[1]JetBrains Research
[2]School of Computation, Information and Technology, Technical University of Munich
tobias.lindenbauer@jetbrains.com

Abstract

Large Language Model (LLM)-based agents solve complex tasks through iterative reasoning, exploration, and tool-use, a process that can result in long, expensive context histories. While state-of-the-art Software Engineering (SE) agents like OpenHands or Cursor use LLM-based summarization to tackle this issue, it is unclear whether the increased complexity offers tangible performance benefits compared to simply omitting older observations. We present a systematic comparison of these approaches with SWE-agent on SWE-bench Verified across five diverse model configurations. Moreover, we show initial evidence of our findings generalizing to the OpenHands agent scaffold. We find that a simple environment Observation Masking strategy halves cost relative to the Raw Agent while matching, and sometimes slightly exceeding, the solve rate of LLM-Summary. Additionally, we introduce a novel hybrid approach that further reduces costs by 7% and 11% compared to just Observation Masking or LLM-Summary, respectively. Our findings raise concerns regarding the trend towards pure LLM-Summary and provide initial evidence of untapped cost reductions by pushing the efficiency-effectiveness frontier. We release code and data for reproducibility.[1][2]

1 Introduction

The ambition to create autonomous agents that can independently handle complex SE tasks is rapidly becoming a reality. These agents, powered by LLMs, typically operate in an iterative loop [39, 29], at each turn they reason about the current state, devise a plan, and execute a tool (e.g., read a file, run tests). The output, or observation, from this tool is then added to the agent's context for the next turn, extending its problem-solving trajectory (Figure 3). This agentic loop acts as a powerful test-time scaling mechanism [25, 17, 32], utilizing the reasoning capabilities [31] of LLMs at each turn while grounding them through environment responses.

However, this iterative context expansion presents a fundamental tradeoff between cost and capability, or effectiveness and efficiency. As the agent's trajectory grows, calls to the LLM become prohibitively expensive due to token-based pricing, and inefficient due to the quadratic attention complexity in the wide-spread Transformer architecture [28]. More critically, even with context windows exceeding 1M tokens, LLMs suffer from the "lost in the middle" problem [16]. While LLMs can process large context windows, they cannot properly make use of relevant information buried within their

[1]Data: https://huggingface.co/datasets/JetBrains-Research/the-complexity-trap
[2]Code: https://github.com/JetBrains-Research/the-complexity-trap

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Deep Learning for Code in the Agentic Era.

vast context [19, 7]. This challenge is acutely amplified in the SE domain, where tool observations are notoriously verbose and noisy [15]. A single action can yield thousands of tokens, whether from reading an entire source file, running a recursive directory listing, or a lengthy test suite log. Concretely, observation tokens make up around 84% of an average SWE-agent turn [35] (Figure 1) in our preliminary experiments (Appendix D.4) on SWE-bench Lite-50 [11, 3]. Due to this, targeting environment observations explicitly provides a strong baseline for LLM-agent context management.

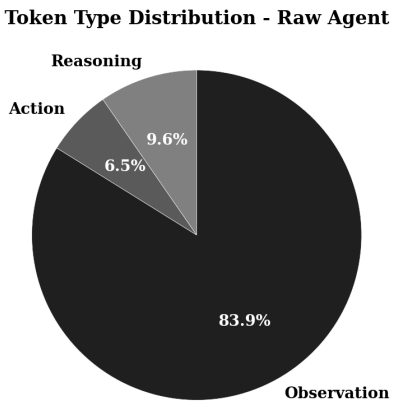

Figure 1: Environment observation tokens dominate the context window of an SE agent's trajectory.

Without any context management strategy targeting cost-efficiency, we find that agent costs can more than double (Table 1), making context management not just an optimization but an economic necessity. In fact, our results show that **any of the discussed management strategies are preferable to none**, as they consistently reduce costs and often improve performance. This raises a critical question: which strategy offers the best trade-off?

A natural baseline approach directly targets verbose environment observations through observation masking strategies that explicitly omit tool outputs. State-of-the-art systems like SWE-agent [35] and SWE-Search [2] have adopted such relatively simple approaches. In parallel, an increasingly popular and more sophisticated solution is to prompt an LLM to perform trajectory summarization, replacing parts of the agent's history with condensed summaries (Figure 3). This approach is adopted by prominent open-source and proprietary SE agents like OpenHands [30] or Cursor [9]. Open-source implementations of both the observation masking and LLM summarization approaches rely on heuristic triggers such as fixed context window sizes or turn thresholds. Thus, the key difference is whether they discard or condense old context. Despite the critical impact of this choice on agent cost and performance, the relative trade-offs between these approaches remain largely unexplored.

In this work, we present a systematic comparison focused on the efficiency of context management strategies. For this, we analyze the performance of representative open-source Observation Masking and LLM-Summary implementations with respect to efficiency (cost in USD) and effectiveness (solve rate on the challenging and industry-standard SWE-bench Verified [5] benchmark). We capitalize these names throughout to denote the specific strategies formally defined in Section 3.1. To enable controlled experimentation across model configurations, we implement LLM-Summary in SWE-agent [35] and adapt the OpenHands' LLM-Summary prompt (Figure 11). We evaluate these strategies within the SWE-agent [35] scaffold and probe for generality on OpenHands [30]. Our experiments span model families, model sizes, licenses (open-weights vs. proprietary), and reasoning regimes (thinking vs. non-thinking).

We find that both Observation Masking and LLM-Summary more than halve the cost compared to the Raw Agent. Furthermore, LLM-Summary is unable to consistently or significantly outperform the simple Observation Masking strategy across all model configurations. We show initial evidence of these findings generalizing to OpenHands. Furthermore, we introduce a novel hybrid approach that further reduces costs by 7% and 11% compared to just Observation Masking or LLM-Summary, respectively. These findings challenge current trends toward pure LLM-Summary and demonstrate that pure LLM-Summary strategies likely leave considerable cost savings untapped.

2 Related Work

Current SE agent research mostly focuses on improving the effectiveness of SE agents by scaling training data [10, 22, 36], selecting the most promising of multiple attempts [22, 10, 40, 2, 1, 33], providing execution-free or execution-based feedback to the agent through critics [2, 22, 10, 40, 24], or enhancing the agent's planning capabilities through explicit search strategies [2, 1]. While these methods improve solve rate, they come at the cost of increased inference costs and thus reduced efficiency. Efficient context management for SE agents, on the other hand, has thus far received

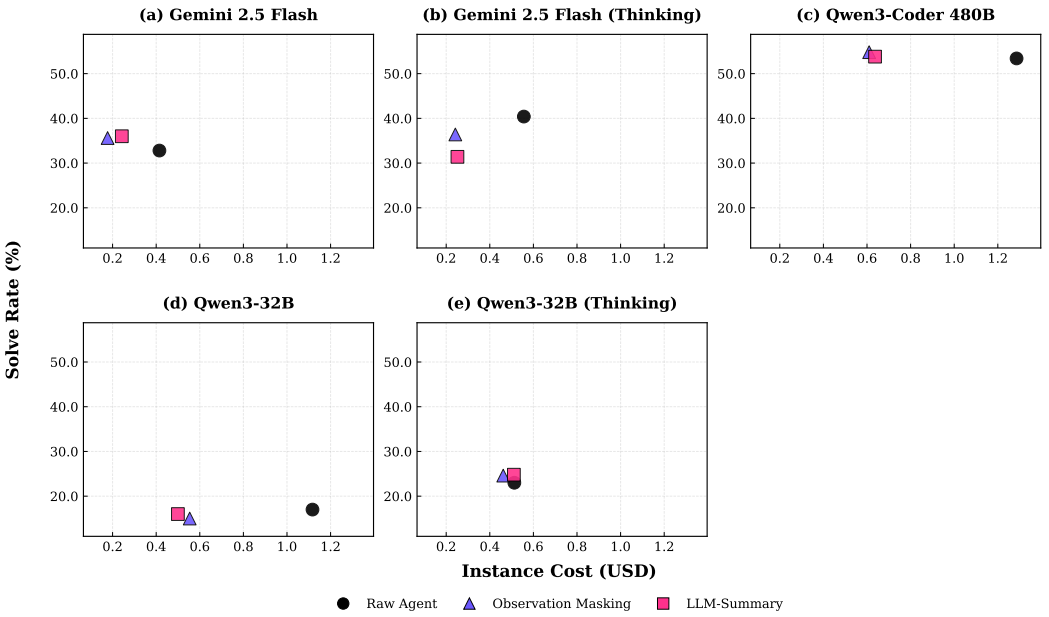

Figure 2: **The effectiveness versus efficiency tradeoff for context management strategies within SWE-agent [35] on SWE-bench Verified [5].** The plot compares solve rate (y-axis, ↑) against the average cost per trajectory (x-axis, ↓) for different model configurations. We test each configuration with three strategies: Raw Agent (baseline, ●), LLM-Summary (■), and Observation Masking (▲). Across all models, the Observation Masking strategy consistently occupies the most efficient frontier, achieving solve rates competitive with, and sometimes superior to, the LLM-Summary strategy. With Qwen3-Coder 480B [27, 8], the best-performing model in our experiments, Observation Masking is not only 52% cheaper than the Raw Agent baseline but also improves on the solve rate by 2.6 %. Moreover, it even **reduces the cost per instance compared to LLM-Summary by $0.03 ($15 across 500 instances)**.

little attention. In this work, we investigate whether complex summarization strategies are necessary for efficient context management in SE agents. For this we experiment with SWE-agent [35] and OpenHands [30].

Recently, several works have conducted in-depth analyses on the effect of the context size on the LLMs performance [16, 7, 19]. These consistently show that LLMs increasingly struggle to effectively utilize the context provided with increased context size. Despite the critical importance of context management for agent performance and deployment costs, existing work treats it as an implementation detail rather than a first-class research question. A recent exception, MEM1 [41], explores dynamic state management for multi-hop QA [37, 13, 12] and web navigation [38] tasks. However, the authors do not compare to omission-based approaches. Furthermore, the benchmarks used result in relatively short trajectories (hundreds of tokens) [41] compared to SE agent trajectories that routinely are orders of magnitude larger [15].

Concurrently with our work, Xiao et al. [34] propose an LLM-Summary variant for efficient SE agent context management. However, they do not compare with an Observation Masking baseline. Most closely related, their "Delete" baseline is an LLM-Summary variant that deletes full turns instead of summarizing them. This baseline does not take advantage of SE trajectories skewing towards environment observations (Figure 1) and breaks the agent reasoning trace across turns, degrading agent downstream performance. Irrespective of this, their results show that this baseline is more efficient than LLM-Summary at a comparable downstream performance. Additionally, Lu et al. [18] investigate using LLM-Summary for tackling context window limitations in RL training of SE and Computer-Use Agent (CUA). Tang et al. [26] on the other hand, achieve impressive performance with Observation Masking for RL training and inference of deep research and CUA agents. This underscores the timeliness of our research and that our findings likely generalize to deep research and CUA.

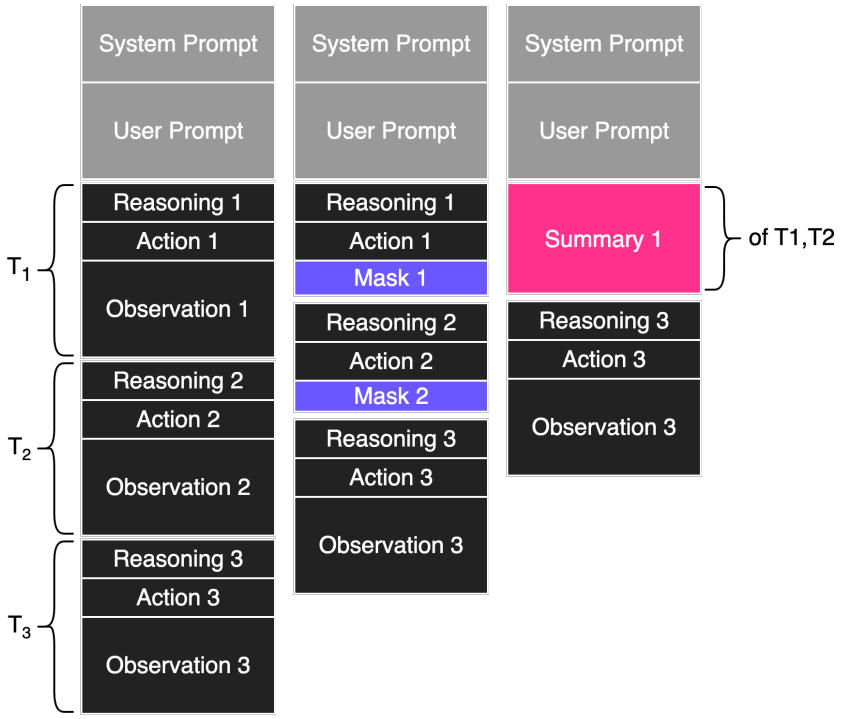

Figure 3: **Overview of the context management strategies evaluated in our work.** Box heights indicate the number of tokens in that portion of a typical trajectory. **(a)** The baseline ReAct-style [39] trajectory, where the context grows with each action-observation pair. **(b) LLM-based summarization** condenses older turns into a running summary and preserving a few recent turns in full (e.g., OpenHands [30]). **(c) Observation masking** replaces observations older than a fixed window of M turns (here, M=1) with a placeholder (e.g., SWE-agent [35]).

3 Experimental Configuration

In our experiments we investigate popular approaches to observation omission and LLM-based summarization through representative open-source implementations: (1) environment Observation Masking through a rolling window [35, 2], and (2) prompt-based LLM-Summary [30, 9]. In the following, we will define our chosen approaches in detail.

3.1 Context Management Strategies

Raw Agent. The agent scaffolds we investigate use either ReAct [39] or CodeAct [29] as an agent framework. In these frameworks, the agent's history, or trajectory, is a sequence of interactions with an environment. We formalize the trajectory at the end of turn $t-1$, denoted τ_{t-1}, as a sequence of tokens:

$$\tau_{t-1} = (o_{sys}, o_{user}, (r_1, a_1, o_1), \ldots, (r_{t-1}, a_{t-1}, o_{t-1})) \tag{1}$$

where o_{sys} and o_{user} are the immutable system and user prompts that initialize the task (Figure 3a). We define a turn $T_i = (r_i, a_i, o_i)$ consisting of reasoning r_i, action a_i and observation o_i as the atomic unit of agent-environment interaction at turn i. This allows us to compactly express τ_{t-1} as:

$$\tau_{t-1} = (o_{sys}, o_{user}, T_1, \ldots, T_{t-1}) \tag{2}$$

At the beginning of turn t, the agent policy π, typically an LLM, conditions on the history τ_{t-1} to generate the next reasoning and action pair: $(r_t, a_t) \sim \pi(\cdot|\tau_{t-1})$. Without any context management strategy, τ grows with each turn, leading to excessive computational costs and eventual context length limitations.

Observation Masking. This strategy manages the context size by selectively condensing past environment observations while preserving the full history of agent reasoning and actions. We define an observation masking function, $f_{mask}(\tau_{t-1}, M)$, that takes the full trajectory τ_{t-1} and an integer window size M as input. The function produces a condensed trajectory, τ'_{t-1}, by replacing environment observations older than the window with a placeholder (Figure 3b).

Formally, given the trajectory $\tau_{t-1} = (o_{sys}, o_{user}, T_1, \ldots, T_{t-1})$, the transformed trajectory is:

$$\tau'_{t-1} = (o_{sys}, o_{user}, (r_1, a_1, o'_1), \ldots, (r_{t-1}, a_{t-1}, o'_{t-1})) \tag{3}$$

where the observation at each turn i, denoted o'_i, is conditionally defined as:

$$o'_i = \begin{cases} p_i & \text{if } i < t - M \\ o_i & \text{if } i \geq t - M \end{cases} \tag{4}$$

Here, p_i is a placeholder text representing the masked observation, such as "Previous 8 lines omitted for brevity.". In the following turns, the agent LLM then conditions on this condensed history τ'_{t-1} to produce (r_t, a_t). This approach, following SWE-agent [35], retains the complete reasoning chain while reducing distant observation fidelity. While this strategy reduces the speed at which the tokens in τ grow, it does not solve the issue of indefinite growth.

LLM-Summary. This strategy uses a "summarizer LLM" to condense the trajectory, which we denote as π'. In contrast to f_{mask}, the goal of this strategy is to maintain salient information of the processed turns. It is controlled by two parameters N and M. N regulates how many turns the agent will accumulate at once, and M regulates how many trailing turns should be left unaltered. We trigger summarization when the accumulated turns since the last summary reach $N + M$.

To help us define this approach, we will introduce two variables, t_{last} and s_{last}. Let t_{last} be the index of the final turn included in the most recent summary ($t_{last} = 0$ at step 0). Let s_{last} be the summary performed at index t_{last} ($s_{last} = o_{user}$ at step 0). Then, we define the summarization as follows. First, we slice the trajectory to obtain the turns eligible for the summarization, containing the last summary s_{last} and all turns between this summary and the M to the last turn $\mathcal{T}_{sum} = (s_{last}, T_{t_{last}+1}, \ldots, T_{t-1-M})$. Then, we generate a new summary s_t by prompting π' with a summary instruction o_{si} and the relevant trajectory slice $\mathcal{T}_{sum}$.

$$\begin{aligned} s_t &\sim \pi'(\cdot | o_{si}, \mathcal{T}_{sum}) \\ t_{last} &= t - 1 - M \end{aligned} \tag{5}$$

Finally, the history provided to the main policy π is reconstructed into a new condensed trajectory, τ'_{t-1}:

$$\tau'_{t-1} = (o_{sys}, o_{user}, s_t, T_{t-M}, \ldots, T_{t-1}) \tag{6}$$

On the next turn, the agent LLM conditions on this new, compact history: $(r_t, a_t) \sim \pi(\cdot | \tau'_{t-1})$. This method ensures the trajectory's growth is not only slowed, but bounded, as older interactions are recursively folded into an evolving summary.

3.2 Experimental Configuration

We conduct a rigorous, comparative study focusing on SWE-agent [35] and probing for generality with OpenHands [30] (`v0.43.0`). Our experiments cover diverse configurations spanning (1) model families, (2) model sizes, (3) model licenses (open-weights vs. proprietary), (4) reasoning regimes (thinking and non-thinking). Concretely, we use Qwen3-32B [27][3] in thinking and non-thinking mode with a context window of 122K tokens using YaRN [23], Qwen3-Coder-480B-A35B-Instruct-FP8 [27, 8][4] with its default context window of 256K tokens, and Gemini 2.5 Flash [6][5] with its default context window of 1M tokens in thinking and non-thinking mode. We conduct all experiments on SWE-bench Verified [5] unless otherwise specified. We run all our experiments on a shared cluster of eight NVIDIA H200 GPUs, each equipped with 141 GB of HBM, and a total of 8 TB local disk storage. For the Qwen-32B [27] models we use two H200 GPUs, and 15 SWE-agent [35]

[3] `https://huggingface.co/Qwen/Qwen3-32B`

[4] `https://huggingface.co/Qwen/Qwen3-Coder-480B-A35B-Instruct-FP8`

[5] API Version: gemini-2.5-flash

inference workers. We choose a conservative number of workers, because we may encounter long trajectories with context sizes > 100K tokens and must account for this in our KV-cache estimates. For Qwen3-Coder 480B [27, 8], we use all eight GPUs and 35 inference workers. We use vLLM [14] to serve the Qwen models on our cluster. In our experiments with Gemini we use eight inference workers on SWE-agent [35] and five on OpenHands [30] due to quota restrictions. For further details see Appendix A.

Our preliminary experiments (Appendix D.4) indicate that the number of turns in the trajectory may influence the behavior of the context management strategies we investigate. Due to this, we experiment with long trajectories and thus set the turn limit to 250 in our experiments unless otherwise specified. For the Observation Masking strategy we use a rolling window size of $M = 10$ in our main experiments, because it resulted in the best performance with SWE-agent [35] in our experiments (Appendix D.1). The LLM-Summary strategy we implement in SWE-agent [35] uses a slightly modified version of the OpenHands-style prompt (Appendix E). In contrast to OpenHands' baseline configuration [30], we summarize 21 turns at once ($N = 21$) and retain only the last ten ($M = 10$) turns. Besides aligning the number of unaltered tail turns for the Observation Masking and LLM-Summary strategies, we also found that $M = 10$ offered the best performance for the LLM-Summary strategy in our experiments (Appendix D.2). For the agent model we use a temperature of 0.8, and for the summary model a temperature of zero. In contrast to experiments with Qwen3-32B [27] thinking, we restrict the thinking budget of Gemini 2.5 Flash to zero or 800 tokens (denoted as *thinking*) due to cost constraints.

4 Main Results

Our main experiments within SWE-agent [35] evaluate three context management strategies, with results summarized in Table 1. The results reveal two central findings that hold robustly across the diverse conditions we tested. First, both Observation Masking and LLM-Summary significantly reduce costs, without significantly reducing solve rate performance. Second, LLM-Summary does not consistently, or significantly outperform Observation Masking on efficiency or effectiveness. For further details see Appendix C.

4.1 The Universal Benefit of Context Management

Our first and most foundational finding, reinforcing the motivation of this study, is that context management is not merely an optimization but a necessity. As shown in Table 1, leaving the agent's context to grow unchecked (the Raw Agent baseline) is consistently the most expensive strategy. In all experimental configurations where trajectories are long enough to benefit from efficient context management, both Observation Masking and LLM-Summary significantly reduce the cost per instance, in most cases by more than 50%. We discuss the outlying behavior of Qwen3-32B (thinking) in Appendix B.

Furthermore, this efficiency does not necessarily come at the cost of performance. In three of our five setups, the most efficient strategy also achieved a higher solve rate than the Raw Agent baseline. This demonstrates that beyond a certain point, more context becomes a liability rather than an asset, aligning with the "lost in the middle" problem [16]. Therefore it is critical to question which approach offers the best trade-off between effectiveness and efficiency.

4.2 Observation Masking: Dominant Efficiency with Minimal Complexity

Having established the clear need for context management, we now turn to the second central finding of our work: the surprising power of simplicity. As we can see in Table 1, in four out of five experimental setups Observation Masking yielded the lowest cost per instance. It achieves this by drastically reducing the number of environment observation tokens processed in each agent turn without incurring the computational overhead of a separate summarization call. This is highly effective, because the agent's context skews heavily towards environment observations in SE (Figure 1). Furthermore, this strategy requires fewer warm-up turns than LLM-Summary ($M = 10$ vs $N + M = 31$). This results in quicker and more robust cost reductions, even on short trajectories (e.g., Qwen3-32B (thinking), see Appendix B).

Table 1: **Comparison of context management strategies with 95% bootstrap confidence intervals.** We report change and significance (†) compared to the *Raw Agent*. We report Solve Rate (effectiveness, ↑) and Instance Cost (efficiency, ↓). For each model, we **boldface** the best-performing context management strategy for each metric. All experiments use SWE-agent [35] on SWE-bench Verified [5]. Further details in Appendix C.

Model	Strategy	Solve Rate (%,↑)	Instance Cost ($,↓)
Qwen3-32B	Raw Agent	17.0 ±3.3	1.12 ±0.18
	Observation Masking	15.0 ±3.1 (-11.8%)	0.55 ±0.09 (-50.9%)†
	LLM-Summary	**16.0** ±3.3 (-5.9%)	**0.50** ±0.07 (-55.4%)†
Qwen3-32B (thinking)	Raw Agent	23.0 ±3.7	0.51 ±0.07
	Observation Masking	24.6 ±3.8 (+7.0%)	**0.46** ±0.05 (-9.8%)
	LLM-Summary	**24.8** ±3.9 (+7.3%)	0.51 ±0.06 (0.0%)
Qwen3-Coder 480B	Raw Agent	53.4 ±4.3	1.29 ±0.26
	Observation Masking	**54.8** ±4.4 (+2.6%)	**0.61** ±0.06 (-52.7%)†
	LLM-Summary	53.8 ±4.2 (+0.7%)	0.64 ±0.06 (-50.4%)†
Gemini 2.5 Flash	Raw Agent	32.8 ±4.1	0.41 ±0.08
	Observation Masking	35.6 ±4.2 (+8.5%)	**0.18** ±0.03 (-56.1%)†
	LLM-Summary	**36.0** ±4.1 (+9.8%)	0.24 ±0.04 (-41.5%)†
Gemini 2.5 Flash (thinking)	Raw Agent	40.4 ±4.3	0.56 ±0.10
	Observation Masking	**36.4** ±4.2 (-9.9%)†	**0.24** ±0.04 (-57.1%)†
	LLM-Summary	31.4 ±4.0 (-22.3%)†	0.25 ±0.05 (-55.4%)†

Notably, this finding holds across model configurations. Furthermore, while $0.03 in cost reductions between LLM-Summary and Observation Masking for Qwen3-Coder 480B seems small, it already amounts to $15 across the entire benchmark. This highlights that even small cost-efficiency gains can have a significant impact on the economic viability or large-scale LLM agent deployments and underscores the need for research on efficient context management.

4.3 Challenging the Need for Complex Summaries

Beyond consistently being the cheapest option, Observation Masking proves to be remarkably effective at maintaining high solve rates. This directly challenges the assumption that complex, semantic summarization is necessary to preserve critical information from an agent's trajectory.

In fact, Observation Masking not only competes with LLM-Summary, but can outperform it. With the Qwen3-Coder 480B [27, 8] model, Observation Masking achieved a solve rate of 54.8%, a slight improvement over the LLM-Summary's 53.8%. Similarly, for Gemini 2.5 Flash [6] (thinking), it outperformed LLM-Summary by five percentage points. In the cases where LLM-Summary did perform better, such as with Gemini 2.5 Flash, the margin was minimal (36.0% vs. 35.6%). This indicates that Observation Masking consistently performs on-par with, or better than the LLM-Summary strategy.

The implication is clear: the most recent context is often sufficient for SE agents. Retaining the entire history, or even a sophisticated summary of it, may not be the most effective use of the model's limited context window and our research budget.

4.4 The Trajectory Elongation Effect

A key question arising from our main results in Table 1 is why the LLM-Summary strategy is less cost-effective than the Observation Masking strategy for all experiments, except Qwen3-32B. Our analysis reveals that this partially stems from an unexpected "trajectory elongation" side-effect of the LLM-Summary context management strategy.

For this, we analyze the distribution of turns per instance in Figure 4. We note that LLM-Summary leads to longer mean trajectory lengths for both Qwen3-Coder 480B and Gemini 2.5 Flash. For Gemini 2.5 Flash the mean trajectory length using LLM-Summary is 52 turns, which is a 15%

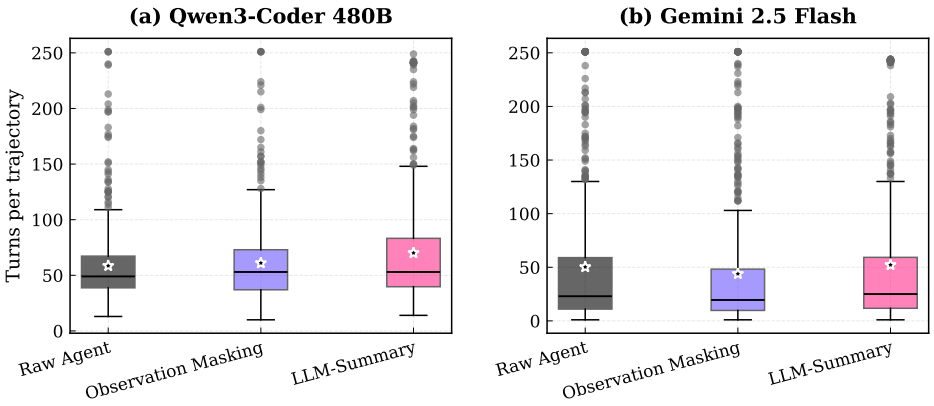

Figure 4: **Impact of context management strategies on trajectory length.** Box plots show the distribution of trajectory lengths (in turns) across different strategies within SWE-agent [35]. LLM-Summary consistently leads to longer trajectories, suggesting they mask failure signals that would otherwise prompt earlier termination. The star indicates the mean trajectory length.

increase over the mean trajectory length of the Observation Masking (44 turns) and 4% over that of the Raw Agent (50 turns). Likewise, for Qwen3-Coder 480B, we observe an increase of the mean trajectory length by 15% compared to the Raw Agent and 13% compared to the Observation Masking strategy.

Notably, this trend of turn elongation translates directly to the efficiency of a strategy observed in Figure 2 and Table 1. The only experiment for which the LLM-Summary strategy proved more cost-efficient is Qwen3-32B. In this case, the Observation Masking strategy, rather than the LLM-Summary strategy led to a 13% increase in mean trajectory length compared to the Raw Agent. This indicates that context summaries act as a reinforcing signal, encouraging the agent to keep going. This in turn results in trajectory elongation that diminishes the efficiency gained through the bounded context achieved by LLM-Summary (Section 3.1).

5 Discussion

In this section we probe the generality of our findings on OpenHands [30], discuss the impact of the LLM summary generation on the cost structure of the LLM-Summary strategy, and introduce a novel hybrid approach combining the strengths of both strategies.

5.1 Probing for Generality With OpenHands

To investigate the generality of our main results across scaffolds, we probe for generality with OpenHands on a 50-instance slice of SWE-bench Verified [5, 3] using Gemini 2.5 Flash [6] (no thinking), with a turn limit of 250, LLM-Summary N=21, M=10, and Observation Masking M=10 and M=58. We present the results Figure 5a.

First, we note that the Observation Masking rolling window size M is an agent-specific hyperparameter that requires tuning. If we simply re-use the optimal value from SWE-agent [35], Observation Masking performance degrades drastically. However, after tuning, we can reproduce our results on this agent scaffold. We hypothesize that we need to tune this hyperparameter due to scaffold-specific implementation details. For example, SWE-agent [35] directly elides retries due to syntax errors from the dialog history. However, OpenHands [30] retains such retry turns. This means we need a larger window size to retain an informative context for this agent.

Overall, this provides initial evidence that our findings generalize across agent scaffolds if the agent's context similarly skews toward environment observations, as typically is the case in SE agents.

5.2 The Costs of Summarization

A closer examination of the cost breakdown reveals that the efficiency gap between strategies stems from two complementary effects. First, the "trajectory elongation" effect discussed in Section 4.4, and second, the costs of generating the summary.

As shown in Table 2, the direct API cost of generating summaries accounts for up to 7.2% of the total instance cost. Importantly, these summarization calls are particularly expensive because each requires processing a unique sequence of turns, limiting cache reuse to the LLM-Summary system prompt (Figure 11). This poses a critical limitation given that, several modern LLM APIs (e.g., Gemini) offer substantially cheaper cache hits than cache misses (up to $10\times$ cheaper). Once we subtract these summarization costs from the total, the efficiency difference between LLM-Summary and Observation Masking largely disappears for most experiments. This indicates that the summarization API calls themselves constitute a substantial portion of the efficiency gap. Nonetheless, the more complex LLM-Summary strategy is still unable to significantly or consistently outperform the remarkably strong Observation Masking strategy on cost-efficiency. This hints at potentially untapped cost savings through underexplored Observation Masking or hybrid approaches.

Table 2: **Mean Instance LLM-Summary Cost per Model.** LLM summary generation API costs explain a portion of the cost-efficiency difference between the LLM-Summary and Observation Masking strategy.

Model	Instance LLM-Summary Cost ($)	Proportional Cost (%)
Qwen3-32B	0.0143	2.86
Qwen3-32B (thinking)	0.0033	0.65
Qwen3-Coder 480B	0.0439	7.20
Gemini 2.5 Flash	0.0161	6.71
Gemini 2.5 Flash (thinking)	0.0131	5.24

5.3 Hybrid: Combined Observation Masking and LLM-Summary

Motivated by the strong individual performances of the context management strategies we cover, both significantly and consistently reducing cost by $> 50\%$, we present a novel hybrid approach in this section. For this, we experiment with the strongest model with respect to solve rate we cover, Qwen3-Coder 480B with SWE-agent [35] on SWE-bench Verified-50 [3] due to cost reasons. We visualize the results in Figure 5b.

Both the trajectory elongation effect (Section 4.4) and the summary generation overhead (Section 5.2) motivate us to treat LLM-Summary as a last resort strategy and defer it as long as possible. However, if we increase N, we increase the number of warm-up turns needed until we observe any effect. During this accumulation phase, we operate under the costly Raw Agent regime. Moreover, we risk not observing any effects on short to medium length trajectories (Appendix B). Using Observation Masking during the turn accumulation phase allows us to combine the strengths of each approach. By increasing N, we defer LLM-Summary and treat this approach as a last resort for bounding the context of long trajectories. At the same time, Observation Masking quickly realizes gains during turn accumulation. Moreover, it does so robustly even on short trajectories.

We set $N = 43$, because at this number of turns the context accumulated under the Observation Masking regime approximately matches the context accumulated under the Raw Agent at $N = 21$ turns ($\approx 30K$ tokens, see Figure 9). To avoid notation clash, we use W for the rolling window size of Observation Masking in the hybrid setup. Overall, we use $N = 43, M = W = 10$ for the hybrid setup. Note that we pass the unmasked context whenever summarizing with LLM-Summary.

Compared to Observation Masking and LLM-Summary, this approach reduces costs by 7% and 11%, respectively. Moreover, it even improves the downstream task performance by 2.6 percent points, pushing the effectiveness-efficiency frontier. This results in expected savings of $20 compared to Observation Masking and $35 compared to LLM-Summary on the full SWE-bench Verified [5] benchmark.

To ablate our hyperparameter choice, we also test the hybrid approach with a naive choice of hyperparameters, disregarding any strategy-specific properties. For this, we simply re-use the

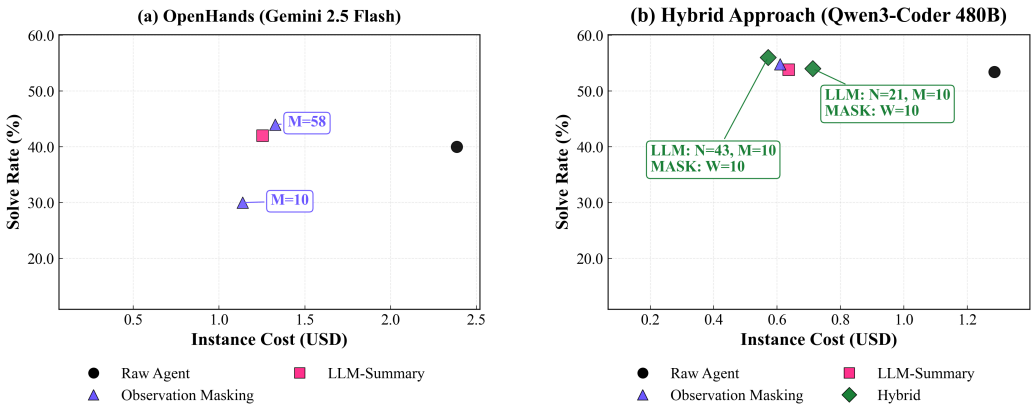

Figure 5: **(a) Probing the generality of our findings with OpenHands [30] on the SWE-bench Verified-50 [3] subset**. After appropriately tuning the rolling window size M to the agent scaffold, Observation Masking again matches the performance of LLM-Summary on both cost and solve rate. **(b) Our novel hybrid Observation Masking and LLM-Summary approach on SWE-bench Verified-50 [3] on the SWE-agent scaffold using Qwen3-Coder 480B.** Effectively combining the strengths of each approach results in a strategy that robustly realizes efficiency gains regardless of trajectory length while benefiting from bounded context on excessively long trajectories. Our hybrid approach yields a slight solve-rate gain of 2.6 percent points compared to the Raw Agent while reducing costs by 7% and 11% compared to Observation Masking and LLM-Summary, respectively.

hyperparameters from the individual approaches $N = 21, M = W = 10$. With this configuration, the hybrid approach actually degrades the systems' overall cost efficiency, due to compounding KV cache inefficiencies, and cost overhead due to LLM-Summary invocations.

6 Limitations

While our study provides a rigorous evaluation of context management strategies, its scope has three main limitations. First, we experiment exclusively within the SE domain, using the SWE-bench [11] benchmark. This domain is characterized by long, verbose tool outputs, a condition that naturally favors the efficiency of Observation Masking. Consequently, our findings on the superiority of this strategy may not generalize to domains where agent-environment interactions are more succinct. Second, all strategies investigated use simple, non-adaptive heuristic triggers. Observation Masking employs a fixed-size rolling window that is agnostic to the relevance or staleness of past observations (e.g., retaining a file's content after it was modified). Similarly, LLM-Summary operates on a fixed turn-based schedule, ignoring semantic boundaries or agent subgoals. Finally, while we provide initial evidence of generalization across agent scaffolds, a more comprehensive investigation may be warranted.

7 Conclusion

This work presents a comprehensive study on context management strategies spanning diverse model configurations and agent scaffolds. We find that efficient context management strategies consistently and significantly reduce system costs by $> 50\%$ without significantly reducing downstream performance. Surprisingly, the popular LLM-Summary strategy is unable to consistently or significantly outperform the simple Observation Masking baseline. This hints at untapped savings potential in modern agentic systems that focus only on LLM-Summary. We empirically validate this hypothesis with our novel hybrid Observation Masking and LLM-Summary strategy that further reduces costs by 7% and 11% compared to Observation Masking and LLM-Summary while improving the downstream task performance. These findings establish the critical need for context management to enable economically feasible, and environmentally sustainable LLM agent deployment. Moreover, it highlights that in the quest for efficient LLM agents, simple solutions can be surprisingly effective.

Acknowledgments and Disclosure of Funding

We would like to thank Calvin Smith for the productive and encouraging discussions on the context management strategies implemented in OpenHands. We would also like to thank Kirill Gelvan for his feedback on early versions of this paper.

References

[1] Vaibhav Aggarwal, Ojasv Kamal, Abhinav Japesh, Zhijing Jin, and Bernhard Schölkopf. DARS: Dynamic action re-sampling to enhance coding agent performance by adaptive tree traversal. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 19808–19855, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.973.

[2] Antonis Antoniades, Albert Örwall, Kexun Zhang, Yuxi Xie, Anirudh Goyal, and William Yang Wang. SWE-search: Enhancing software agents with monte carlo tree search and iterative refinement. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=G7sIFXugTX.

[3] Ibragim Badertdinov, Maria Trofimova, Yuri Anapolskiy, Sergey Abramov, Karina Zainullina, Alexander Golubev, Sergey Polezhaev, Daria Litvintseva, Simon Karasik, Filipp Fisin, Sergey Skvortsov, Maxim Nekrashevich, Anton Shevtsov, and Boris Yangel. Scaling data collection for training software engineering agents. *Nebius blog*, 2024.

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

[5] Neil Chowdhury, James Aung, Chan Jun Shern, Oliver Jaffe, Dane Sherburn, Giulio Starace, Evan Mays, Rachel Dias, Marwan Aljubeh, Mia Glaese, Carlos E. Jimenez, John Yang, Leyton Ho, Tejal Patwardhan, Kevin Liu, and Aleksander Madry. Introducing swe-bench verified, 2024. URL https://openai.com/index/introducing-swe-bench-verified/. Accessed on March 12, 2025.

[6] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, et al. Technical report, Google DeepMind, 2025.

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

[8] Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

[9] Anysphere Inc. *Summarization: Context management for long conversations in chat*. URL https://docs.cursor.com/en/agent/chat/summarization. Accessed Aug. 7, 2025.

[10] Naman Jain, Jaskirat Singh, Manish Shetty, Liang Zheng, Koushik Sen, and Ion Stoica. R2E-Gym: Procedural Environments and Hybrid Verifiers for Scaling Open-Weights SWE Agents, 2025.

[11] 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.

[12] Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

[13] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 2019.

[14] 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.

[15] Tobias Lindenbauer, Georg Groh, and Hinrich Schuetze. From knowledge to noise: CTIM-rover and the pitfalls of episodic memory in software engineering agents. In *Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025)*, pages 411–427, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.realm-1.30.

[16] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. 12:157–173, 2024. doi: 10.1162/tacl_a_00638.

[17] Runze Liu, Junqi Gao, Jian Zhao, Kaiyan Zhang, Xiu Li, Biqing Qi, Wanli Ouyang, and Bowen Zhou. Can 1b LLM surpass 405b LLM? rethinking compute-optimal test-time scaling.

[18] Miao Lu, Weiwei Sun, Weihua Du, Zhan Ling, Xuesong Yao, Kang Liu, and Jiecao Chen. Scaling LLM multi-turn RL with end-to-end summarization-based context management. URL `http://arxiv.org/abs/2510.06727`.

[19] Ali Modarressi, Hanieh Deilamsalehy, Franck Dernoncourt, Trung Bui, Ryan A. Rossi, Seunghyun Yoon, and Hinrich Schütze. NoLiMa: Long-context evaluation beyond literal matching.

[20] OpenAI, Ananya Kumar, Jiahui Yu, John Hallman, et al. Introducing gpt-4.1 in the api. Technical report, OpenAI, April 2025. URL `https://openai.com/index/gpt-4-1/`.

[21] OpenAI et al. GPT-4 Technical Report, March 2024. URL `http://arxiv.org/abs/2303.08774`. arXiv:2303.08774 [cs.CL].

[22] Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training Software Engineering Agents and Verifiers with SWE-Gym, December 2024. URL `http://arxiv.org/abs/2412.21139`. arXiv:2412.21139 [cs].

[23] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models.

[24] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652, 2023. URL `https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf`.

[25] Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2408.03314.

[26] Qiaoyu Tang, Hao Xiang, Le Yu, Bowen Yu, Yaojie Lu, Xianpei Han, Le Sun, WenJuan Zhang, Pengbo Wang, Shixuan Liu, Zhenru Zhang, Jianhong Tu, Hongyu Lin, and Junyang Lin. Beyond turn limits: Training deep search agents with dynamic context window. URL `http://arxiv.org/abs/2510.08276`.

[27] Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[29] Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable Code Actions Elicit Better LLM Agents. In *Proceedings of the 41st International Conference on Machine Learning*, pages 50208–50232, July 2024. URL https://proceedings.mlr.press/v235/wang24h.html.

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

[31] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Proceedings of the 36th International Conference on Machine Learning.*, 2022. doi: 10.48550/arXiv.2201.11903.

[32] Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Scaling inference computation: Compute-optimal inference for problem-solving with language models. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*, 2024. URL https://openreview.net/forum?id=j7DZWSc8qu.

[33] Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying LLM-based software engineering agents.

[34] Yuan-An Xiao, Pengfei Gao, Chao Peng, and Yingfei Xiong. Improving the efficiency of LLM agent systems through trajectory reduction. URL http://arxiv.org/abs/2509.23586.

[35] John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://doi.org/10.48550/arXiv.2405.1579.

[36] John Yang, Kilian Leret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling Data for Software Engineering Agents, April 2025. arXiv:2504.21798 [cs].

[37] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering.

[38] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents.

[39] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

[40] Kexun Zhang, Weiran Yao, Zuxin Liu, Yihao Feng, Zhiwei Liu, Rithesh R N, Tian Lan, Lei Li, Renze Lou, Jiacheng Xu, Bo Pang, Yingbo Zhou, Shelby Heinecke, Silvio Savarese, Huan Wang, and Caiming Xiong. Diversity empowers intelligence: Integrating expertise of software engineering agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=cKlzKs3Nnb.

[41] Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents.

A Experimental Configuration

To compute the instance cost reported in our experiments, we distinguish between proprietary models we access through the API, and models we host locally on our infrastructure. For the Gemini [6] experiments, which we access through the Vertex AI API, we report the cost returned by the API. For Qwen [27, 8] experiments, we self-host the models as described in Section 3.2. To compute the cost, we use the observed per-turn token usage and post-hoc compute the cost based on the official Alibaba API pricing[6]. Note that in contrast to the Vertex AI platform, the official Alibaba API pricing for the Qwen3-32B model, does not distinguish between cache hit and miss input tokens. This leads to an inflated cost of the Qwen3-32B experiments. We refer readers interested in the cost structure of our experiments under different pricing schemes to out HuggingFace dataset[7], in which we release all experimental data for our main experiments and our repository[8].

B Short Trajectories in Qwen3-32B (thinking): Hypotheses and Implications

The Qwen3-32B (thinking) configuration exhibits an around 50% shorter median trajectory length compared to the Qwen3-32B configuration. Given that the only change between the two configurations is that we enable model thinking, this is surprising. In the following, we detail our investigation of this outlier result and discuss its implications on the interpretation of our findings.

Configuration validation. We verified that the configuration and dataset are correct for these two experimental configurations across all context management strategies. Additionally, we did not observe deviations in the distribution of instance exit statuses between the two configurations, besides the slightly improved performance of the model under the thinking regime (**??**).

Qualitative analysis. As we did not identify a misconfiguration, we further performed a qualitative analysis of the same 20 (4%) trajectories across the Qwen3-32B experimental configurations. Here, we noticed that both models sometimes struggle with following the function calling format of the agent scaffold. However, as discussed above, we did not observe suspicious deviations in the number of exits due to function calling errors in either configuration.

Implications. Because we could not identify any misconfiguration or otherwise suspicious behavior as the reason for this outlier result with the Qwen3-32B (thinking) configuration, we must assume that it is valid. Thus, we now discuss the implications of this result on the interpretation of our overall findings. First, recall that this configuration resulted in short trajectories. However, both of our context management strategies need a number of warm-up turns, before they start modifying the trajectory and thus reducing cost. For LLM-Summary we require $N + M = 31$ turns before we produce the first summary with our hyperparameters. This means the trajectories with a median length of 15 turns are far too short to realize efficiency gains from this context management strategy. Observation Masking on the other hand, starts masking observations after $M = 10$ turns. This means we expect to see an effect when using this strategy on shorter trajectories, however it may be muted. This exactly matches the empirical behavior observed in **??**. Therefore, we attribute the insignificant cost savings in the Qwen3-32B (thinking) configuration to the shorter trajectory lengths, rather than fundamental issues with our context management strategies. This interpretation is further supported by the fact that even in this unfavorable setting, Observation Masking reduces cost by $\approx 10\%$, and both strategies still result in stable downstream performance.

[6] `https://www.alibabacloud.com/help/en/model-studio/models#16ff9753e1ctz`

[7] Data: `https://huggingface.co/datasets/JetBrains-Research/the-complexity-trap`

[8] Code: `https://github.com/JetBrains-Research/the-complexity-trap`

C Detailed Main Results

In this section, we provide further data on which we base our confidence intervals and significance indicators in Table 1. Table 3 presents the full asymmetric confidence intervals underlying our main results. The symmetric intervals in Table 1 are derived by averaging the asymmetric bounds: e.g., $17.0^{+3.4}_{-3.2}$ yields 17.0 ± 3.3.

Table 3: Comparison of context management strategies with 95% bootstrap confidence intervals, showing asymmetry. We use † to indicate significance compared to the Raw Agent. We report Solve Rate (effectiveness, ↑) and Instance Cost (efficiency, ↓). For each model, we **boldface** the best-performing context management strategy for each metric (relative to the Raw Agent baseline). Change is reported relative to the *Raw Agent* baseline. All experiments use SWE-agent [35] on SWE-bench Verified [5].

Model	Strategy	Solve Rate (%,↑)	Instance Cost ($,↓)
Qwen3-32B	Raw Agent	$17.0^{+3.4}_{-3.2}$	$1.12^{+0.18}_{-0.17}$
	Observation Masking	$15.0^{+3.2}_{-3.0}$ (-11.8%)	$0.55^{+0.09}_{-0.08}$ (-50.9%)†
	LLM-Summary	**16.0**$^{+3.4}_{-3.2}$ (-5.9%)	**0.50**$^{+0.07}_{-0.06}$ (-55.4%)†
Qwen3-32B (thinking)	Raw Agent	$23.0^{+3.8}_{-3.6}$	$0.51^{+0.07}_{-0.06}$
	Observation Masking	$24.6^{+3.8}_{-3.8}$ (+7.0%)	**0.46**$^{+0.05}_{-0.05}$ (-9.8%)
	LLM-Summary	**24.8**$^{+4.0}_{-3.8}$ (+7.3%)	$0.51^{+0.06}_{-0.06}$ (0.0%)
Qwen3-Coder 480B	Raw Agent	$53.4^{+4.4}_{-4.2}$	$1.29^{+0.28}_{-0.24}$
	Observation Masking	**54.8**$^{+4.4}_{-4.4}$ (+2.6%)	**0.61**$^{+0.06}_{-0.05}$ (-52.7%)†
	LLM-Summary	$53.8^{+4.2}_{-4.2}$ (+0.7%)	$0.64^{+0.06}_{-0.05}$ (-50.4%)†
Gemini 2.5 Flash	Raw Agent	$32.8^{+4.2}_{-4.0}$	$0.41^{+0.08}_{-0.07}$
	Observation Masking	$35.6^{+4.2}_{-4.2}$ (+8.5%)	**0.18**$^{+0.03}_{-0.02}$ (-56.1%)†
	LLM-Summary	**36.0**$^{+4.2}_{-4.0}$ (+9.8%)	$0.24^{+0.04}_{-0.04}$ (-41.5%)†
Gemini 2.5 Flash (thinking)	Raw Agent	$40.4^{+4.2}_{-4.4}$	$0.56^{+0.10}_{-0.10}$
	Observation Masking	**36.4**$^{+4.2}_{-4.2}$ (-9.9%) †	**0.24**$^{+0.04}_{-0.03}$ (-57.1%)†
	LLM-Summary	$31.4^{+4.0}_{-4.0}$ (-22.3%)†	$0.25^{+0.05}_{-0.04}$ (-55.4%)†

C.1 Statistical Analysis

We assess significance using paired nonparametric bootstrap with $B = 10{,}000$ replicates and show detailed results in Table 4. For each model-strategy pair, we compute the paired difference $\Delta =$ mean(strategy) − mean(raw) on the same $n = 500$ instances, preserving instance-level correlations. We report:

- 95% percentile confidence intervals
- Two-sided p-values: $p = 2 \times \min(\Pr(\Delta^* \geq 0), \Pr(\Delta^* \leq 0))$
- Significance markers (†) when $p < 0.05$

Table 4 provides the complete bootstrap statistics. Note that p-values of 0.0000 indicate no sign-crossing across all bootstrap replicates (resolution $\leq 10^{-4}$).

D Additional Studies

For the critic-enhanced summarizer in Figure 7, we experiment on SWE-bench Lite-50 [3]. For the sensitivity to the rolling window size M of the Observation Masking strategy and the configurations of the LLM-Summary strategy, we show our results on a randomly sampled 150-instance subset of SWE-bench Verified [5] that we release with our code. We conduct these studies with GPT-4.1-mini [21].

Table 4: Paired bootstrap differences vs. Raw Agent with 95% percentile CIs and two-sided bootstrap p-values (B=10,000). Δ Solve Rate is reported in percentage points (pp), Δ Mean Cost in dollars per instance. Negative cost differences indicate cost savings. All rows use n=500 common instances per model. We use † to indicate significance compared to the Raw Agent.

Model	Strategy	Δ Solve Rate (pp) [lo, hi]	p	Δ Mean Cost ($) [lo, hi]	p
Gemini 2.5 Flash	Observation Masking	2.8 [-0.8, 6.4]	0.1504	-0.2377 [-0.3202, -0.1614]	0.0000†
	LLM-Summary	3.2 [-0.4, 7.0]	0.0948	-0.1725 [-0.2579, -0.0936]	0.0000†
Gemini 2.5 Flash (thinking)	Observation Masking	-4.0 [-7.8, -0.2]	0.0406†	-0.3143 [-0.4096, -0.2245]	0.0000†
	LLM-Summary	-9.0 [-13.0, -5.2]	0.0000†	-0.3046 [-0.4074, -0.2043]	0.0000†
Qwen3-Coder 480B	Observation Masking	1.4 [-1.6, 4.4]	0.3856	-0.6762 [-0.9320, -0.4518]	0.0000†
	LLM-Summary	0.4 [-3.0, 3.8]	0.8736	-0.6491 [-0.9048, -0.4263]	0.0000†
Qwen3-32B	Observation Masking	-2.0 [-5.0, 1.0]	0.2086	-0.5632 [-0.7479, -0.3817]	0.0000†
	LLM-Summary	-1.0 [-4.6, 2.6]	0.6192	-0.6174 [-0.7904, -0.4454]	0.0000†
Qwen3-32B (thinking)	Observation Masking	1.6 [-2.0, 5.2]	0.3980	-0.0510 [-0.1255, 0.0187]	0.1586
	LLM-Summary	1.8 [-1.8, 5.4]	0.3420	-0.0021 [-0.0785, 0.0741]	0.9370

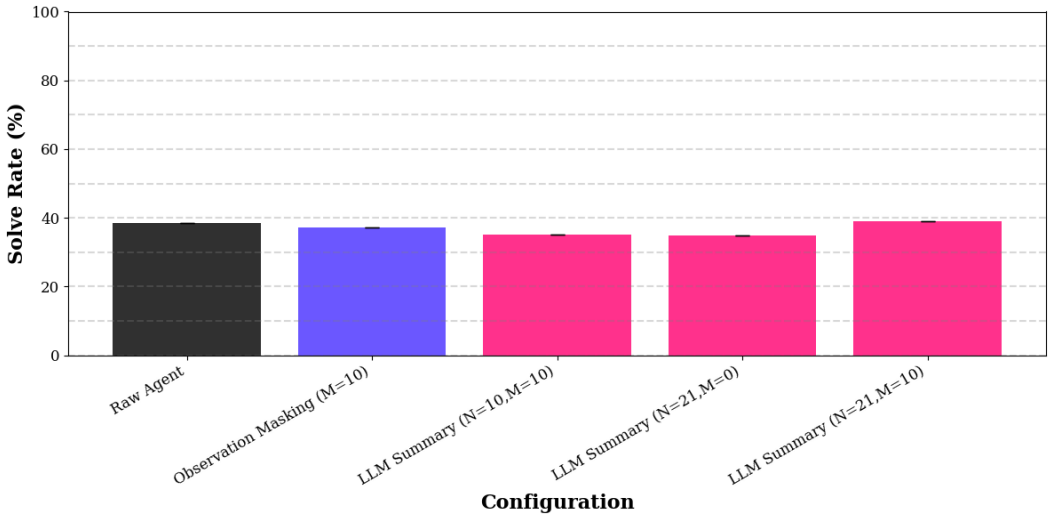

Figure 6: Downstream task performance of a single experiment on a randomly generated 150-sample subset of SWE-bench Verified [5] across various different configuration combinations with respect to the tail length M. We find that a larger summarization window compared to the tail length improves performance.

D.1 Observation Masking Configuration

We experiment with the rolling window size M of the Observation Masking strategy. In Figure 10 we can see that the performance of the strategy peaks at $M = 10$, before falling again when further increasing the window size to $M = 20$. Thus, we use this configuration of the Observation Masking strategy for our main experiments.

D.2 LLM-Summary Configuration

In Figure 6 we show the solve rate of different experimental configurations for LLM-Summary in addition to those of our baselines. We find that using tail turns $M > 0$ improves downstream performance. Furthermore, in contrast to the 50-50 split between turns to summarize and tail turns that OpenHands [30] uses, we find that summarizing more turns at once improves the solve rate. We thus proceed with $N = 21, M = 10$ for our main experiments.

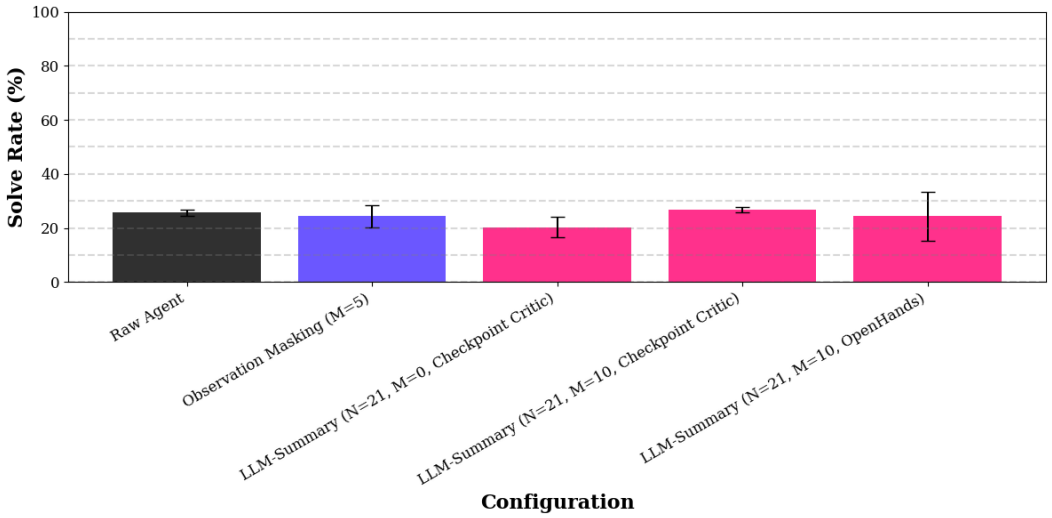

Figure 7: Downstream task performance a randomly generated 150-sample subset of SWE-bench Verified [5] comparing the prompt in Appendix E with the joint critic-summarization prompt presented in this section. We find that simply prompting the model to include feedback in its summaries does not improve the solve rate and further increases the cost.

D.3 Critic-Enhanced LLM-Summary

A natural follow-up question is whether the LLM-Summary strategy could be improved by making the summarization process more intelligent. We explore this by enhancing our LLM-Summary strategy with execution-free feedback, a technique that has shown promise in scaling test-time compute for SE agents [22, 10, 40, 2]. In this approach, the LLM simultaneously generates a summary and critical analysis of the trajectory, incorporating both into the compressed context. This is akin to providing reflections within a single rollout, instead of across multiple rollouts [24].

In comparison to the modified OpenHands prompt in Figure 11, we frame the task as generating a checkpoint instead of a summary and prompt the model to reflect on the turns to summarize. In doing so, we aim to encourage the model to generate an output that helps the agent adjust its solution path during an attempt and avoid overly grounding it in previous, potentially suboptimal or even flawed, turns through a plain summary.

To elicit meaningful reflections, we prompt the LLM with guiding questions that it could reflect and provide insights on. These questions assess whether the agent is stuck or looping, aligned with the initial problem statement, reflect on the agent's high-level solution approach with respect to the turns to summarize. Additionally, we provide few-shot examples to further guide the agents toward generating meaningful and actionable reflections [4]. Finally, as we did in the OpenHands-style prompt (Figure 11), we provide the previous summary, or problem statement if none is available, and the turns to summarize to the model.

Testing on 150 samples from SWE-bench Verified [5] using SWE-agent [35], this critic-enhanced approach using the prompt presented in Figures 12 to 14 showed no improvement in solve rate over standard LLM-Summary. More concerning, we observed exacerbated trajectory elongation patterns, with critic-enhanced runs producing even longer trajectories than standard summarization. This is perhaps unsurprising: the critic's reflections naturally encourage the agent to explore alternative solution paths, try additional debugging strategies, or reconsider its approach, all of which translate to more turns, thus driving cost and reducing efficiency gains.

This finding reinforces our central insight about trajectory elongation. While execution-free feedback aims to improve agent decision-making, it paradoxically increases computational costs by extending exploration. The critic's guidance, rather than helping the agent efficiently recognize dead ends, provides additional avenues to pursue, further delaying termination. Furthermore, this increased cost, does not lead to increased downstream performance. This suggests that effective memory systems for

AI agents require fundamental rethinking: simply adding more sophisticated feedback to summaries may compound rather than solve the efficiency challenges we identify.

D.4 Behavior of the Covered Context Management Strategies Across Turns

In Figure 8, we show preliminary experimental results using the trajectory management strategies introduced in Section 3.1 on SWE-bench Lite-50 [3] with GPT-4.1-mini [20]. Here, we use a rolling window size of $M = 5$, following SWE-agent [35] and $N = 21, M = 10$ for the LLM-based approach paired with a slightly modified version of OpenHand's prompt [30] (see Appendix E). We observe that the Observation Masking strategy poses a strong baseline, consistently performing equally or better on the downstream task on SWE-bench Lite-50 [3] than the LLM-Summary approach despite using being much simpler.

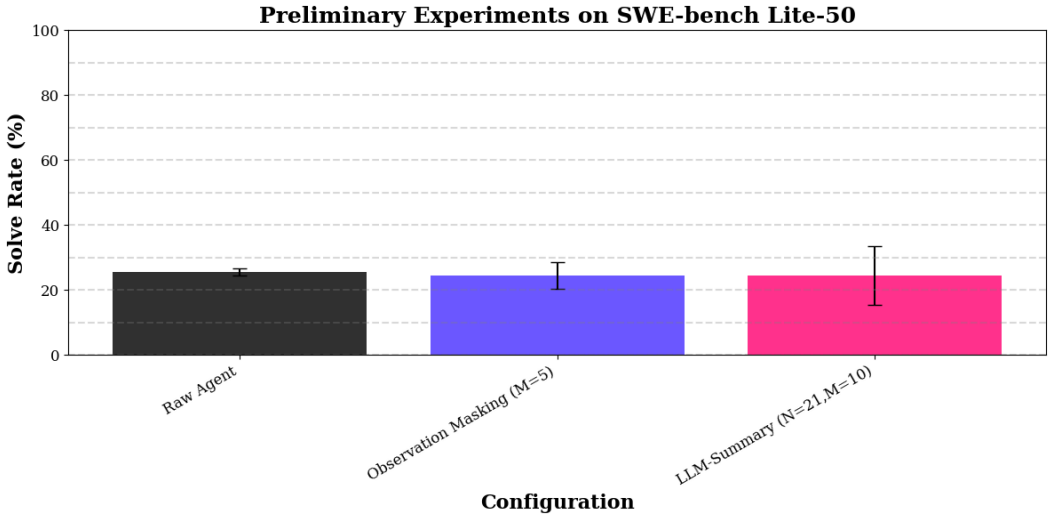

Figure 8: Downstream task performance on SWE-bench Lite-50[3] across the context management strategies we cover in our study. Bars represent the mean across three experiments, the error bars show the standard deviations. Surprisingly, f_{RW} performs on par with the LLM-based strategy using more compute.

To investigate why this is the case, and uncover potential scenarios in which the LLM-Summary strategy may be beneficial, we analyze the behavior of these strategies across turns in Figure 9. The solid colored lines are the empirically observed results using SWE-agent [35]. For each trajectory management strategy, we run three experiments, yielding 150 trajectories total. To visualize these data, we use the micro-averaged mean for each turn. The dashed lines indicate the empirically grounded simulated behavior. To generate the data for these simulated trajectories, we compute the mean token consumption per token type across all experimental data available for the raw agent:

$$\bar{x} = \frac{1}{T_{total}} \sum_{i=1}^{3} \sum_{j=1}^{50} \sum_{k=1}^{T_{local}} x_{ijk} \quad \text{where} \quad x \in \{r, a, o\} \tag{7}$$

where T_{total} is the total number of turns T we observed across all instances and experiments and T_{local} is the number of turns of a single trajectory. We then generate a turn $T_{sim} = (\bar{r}, \bar{a}, \bar{o})$ using placeholder tokens. By repeatedly appending T_{sim} we generate a simulated agent trajectory $\tau_{sim} = (T_{sim}, \ldots, T_{sim})$ of arbitrary length. To generate the data for the simulated LLM-Summary and Observation Masking context management trajectories, we apply these strategies to τ_{sim}. This allows us to study the expected behavior of these trajectory management strategies up to a large number of turns.

Figures 9 a, b and c show that our experimental data match the expected behaviour of simulated trends closely. Surprisingly, we find that the f_{RW} **is competitive with the the LLM-based strategy on cost and even outperforms it on context compression**.

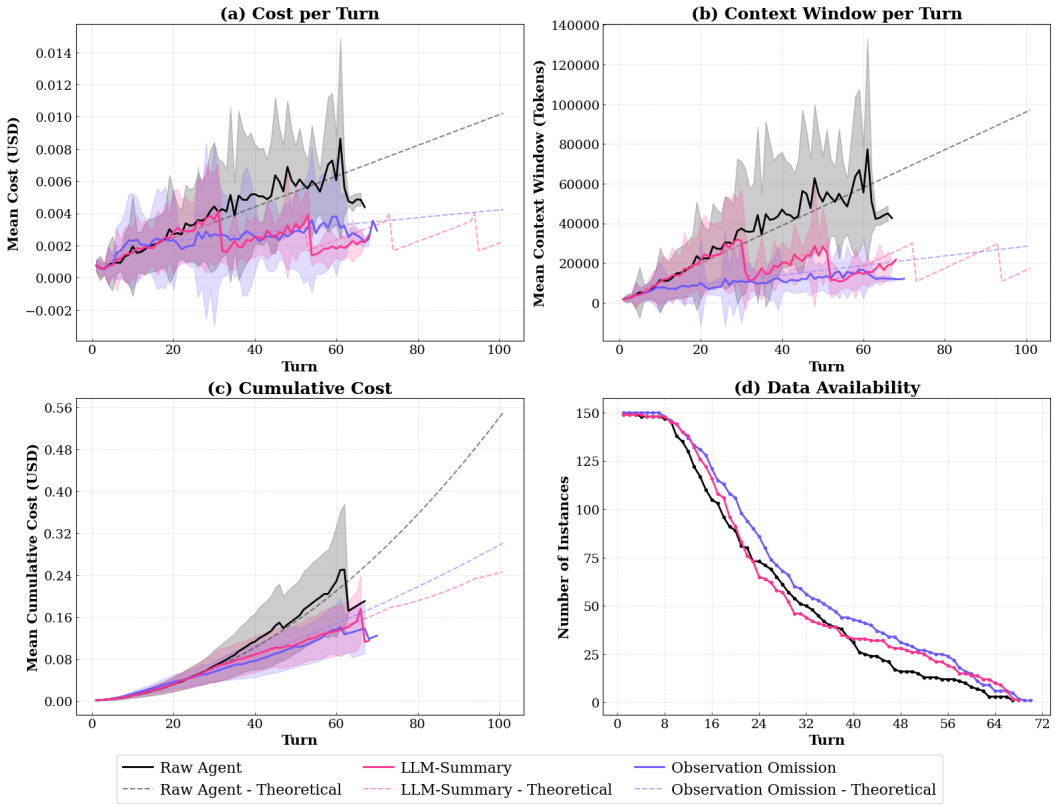

Figure 9: **Preliminary experimental results motivating our study.** Dashed lines show expected behavior based on mean token counts per turn type observed in the raw agent trajectories. We micro-average all results with standard deviation shown as shaded regions. **(a),(b), and (c)** The observed data closely match the simulated effects of applying either trajectory management strategy to the simulated raw agent trajectory. The effects of the LLM and Observation Masking approach on cost and context window size overlap at lower turn numbers. Due to the bounding of the context-window we expect the LLM approach to be especially effective on long trajectories. **(d)** With an increasing number of turns, our empiric data becomes increasingly sparse.

f_{RW} is a strong baseline, due to the distribution of tokens across the types r, a, o. In Figure 1 we plot the share of token types in T_{sim}. The environment observation tokens o overwhelmingly dominate the composition of T_{sim}, contributing $\approx 84\%$. Thus targeting this token type is extremely effective.

We can see this in effect in Figures 9a, and b. While the cost of the two context management strategies is similar, due to the worse cache behavior of f_{RW}, f_{RW} offers superior compression especially at lower turn numbers. Looking at our simulations on the other hand, we expect the LLM-based approach to start outperforming f_{RW} on longer trajectories because it bounds the maximum context size in a fuzzy manner, resulting in a saw-function for both the cost and context window size. This motivates us to set the turn limit in our main experiments to 250.

E LLM Summary Prompts

We share our prompt template for summary generation in Figure 11. Compared to OpenHands [30], we remove the part of the prompt that aims to handle summary generation for tasks outside the SE domain, since our work is purely focused on the SE domain. In addition to the system prompt shown in Figure 11, we provide a joint critic-summarization prompt in Figures 12 to 14. We discuss the effects of generating execution-free in Appendix D.3 and Section 5.

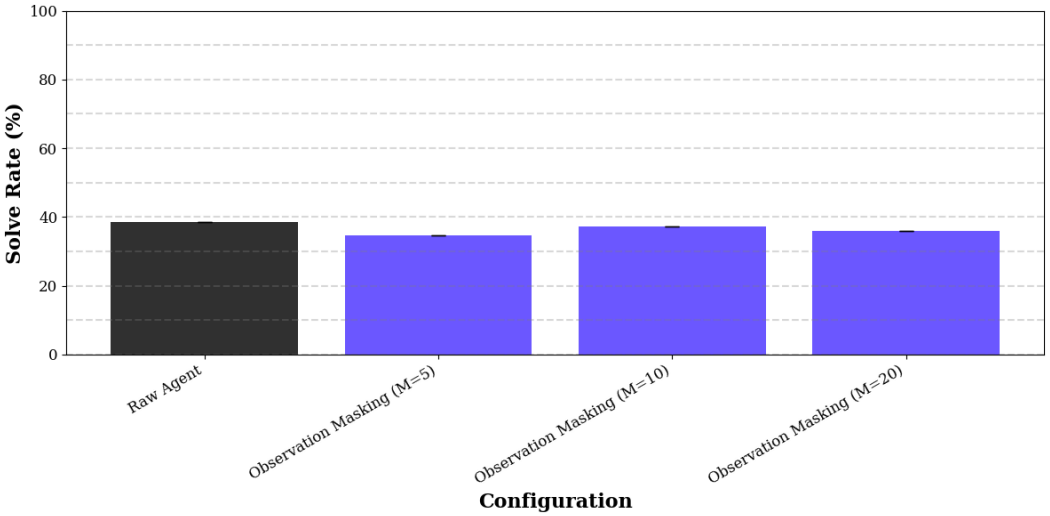

Figure 10: Downstream task performance of a single experiment on a randomly generated 150-sample subset of SWE-bench Verified [5] across different context window sizes. We find that $M = 10$ yields optimal performance.

LLM-Summary Prompt

```
You are maintaining a context-aware state summary for an interactive agent.
You will be given a list of events corresponding to actions taken by the
agent, and the most recent previous summary if one exists. Track:

USER_CONTEXT: (Preserve essential user requirements, goals, and
clarifications in concise form)

COMPLETED: (Tasks completed so far, with brief results)
PENDING: (Tasks that still need to be done)
CURRENT_STATE: (Current variables, data structures, or relevant state)

For code-specific tasks, also include:
CODE_STATE: (File paths, function signatures, data structures)
TESTS: (Failing cases, error messages, outputs)
CHANGES: (Code edits, variable updates)
DEPS: (Dependencies, imports, external calls)
VERSION_CONTROL_STATUS: (Repository state, current branch, PR status,
commit history)

PRIORITIZE:
1. Adapt tracking format to match the actual task type
2. Capture key user requirements and goals
3. Distinguish between completed and pending tasks
4. Keep all sections concise and relevant

SKIP: Tracking irrelevant details for the current task type

Example formats:

For code tasks:
USER_CONTEXT: Fix FITS card float representation issue
COMPLETED: Modified mod_float() in card.py, all tests passing
PENDING: Create PR, update documentation
CODE_STATE: mod_float() in card.py updated
TESTS: test_format() passed
CHANGES: str(val) replaces f"{val:.16G}"
DEPS: None modified
VERSION_CONTROL_STATUS: Branch: fix-float-precision, Latest commit: a1b2c3d

<PREVIOUS_SUMMARY>
...
</PREVIOUS_SUMMARY>
<TURN-0>
...
</TURN-0>
...
<TURN-20>
...
</TURN-20>
```

Figure 11: The LLM-Summary prompt we use in SWE-agent [35] is a slightly modified version of the OpenHands LLM-Summary system prompt [30]. Additionally we pass the previous summary and the turns to summarize. If no previous summary is available we instead pass the task problem statement as initial context for the summary generation.

Joint Critic-Summary Prompt (Part 1)

```
You are maintaining a context-aware state checkpoint for an interactive agent
working on software engineering tasks (specifically, bug fixes), assessing the
agents progress toward completing the task, and offering suggestions and
guidance if it is not on track.
You will be given a list of turns corresponding to actions taken by the agent
and their resulting observations, and the most recent previous checkpoint if
one exists. You must proceed in the following two phases:
1. <CHECKPOINT>
2. <REFLECTIONS>

A <CHECKPOINT> should capture the current repository state and the agent's
progress towards completing the task. It consists of:
USER_CONTEXT: (Preserve essential user requirements, goals, and clarifications
based on findings, previous checkpoints, and the initial problem statement in
concise form)
CODE_STATE: (File paths, function signatures, data structures followed by
their current state)
TESTS: (Failing cases, error messages, outputs)
CHANGES: (Code edits, variable updates)
DEPS: (Dependencies, imports, external calls)

PRIORITIZE:
1. Capture key requirements from the initial issue description or previous
checkpoints and reflections
2. Keep all sections concise and relevant to fixing the issue
3. Focus on information that quantifies the agent's progress towards a solution

Next you must reflect on the agent's progres in the below to turns to
generate a set of <REFLECTIONS>. Here are some aspects to consider
when generating the <REFLECTIONS>:
- Are the agent's actions still aligned with the initial user requirement?
- Is the agent making progress?
- Is it stuck in a loop or repeatedly carrying out the same actions?
- Can you identify any problematic patterns in the agent's actions?
- Did the agent follow the issue description or your previous feedback?
    If so, why did or didn't it make meaningful progress?
- Which critical piece of information might help the agent get back on
    track?
- What has the agent not tried so far?

Key requirements for the <REFLECTIONS>:
1. Avoid reporting nitpicks as they may confuse the agent.
2. Your reflections should be diverse with respect to any available previous
reflections.
3. Provide up to 2 reflections total. Reflections are mutually exclusive.
4. Each reflection should identify one distinct problem and may include one
or two fixes for that problem.
5. Limit yourself to the most critical issues that are blocking progress.

When generating the <REFLECTIONS>, follow the format below:
<REFLECTIONS>
Problem-A: (a detailed description of a problem the agent is facing)
Fix-A.1: (proposed solution, guidance or hint for overcoming the problem
including the rationale for it)
</REFLECTIONS>
```

Figure 12: Part 1 of our joint critic and summarization LLM-Summary prompt. Compared to the LLM-Summary prompt we use in our main experiments (Figure 11), we also prompt the LLM to act as execution-free critic regarding the turns it is summarizing.

Joint Critic-Summary Prompt (Part 2)

```
Example output and format:
<CHECKPOINT>
USER_CONTEXT: Fix failing authentication in REST API. Users report
"Invalid token" errors after ~30 minutes of activity. The API should
maintain user sessions properly with JWT tokens that expire after 1 hour.
CODE_STATE:
1. api/auth_middleware.py: validate_token() MODIFIED with logging.
2. api/auth_utils.py: refresh_token() MODIFIED to auto-refresh at 45 min.
3. config.py: JWT_EXPIRATION unchanged at 3600.
TESTS:
1. tests/test_auth_integration.py::test_long_session: FAILING - "Token
expired at 32 minutes"
2. tests/test_auth_integration.py::test_token_refresh: PASSING
3. tests/test_auth_unit.py: ALL PASSING
CHANGES:
1. auth_utils.py: Added auto-refresh logic when token age > 2700 seconds.
2. auth_middleware.py: Added debug logging for token validation steps.
DEPS: PyJWT==2.4.0, python-jose==3.3.0 (both imported)
</CHECKPOINT>
<REFLECTIONS>
Problem-A: Agent has spent 6 turns modifying the token refresh logic
and adding complex auto-refresh mechanisms, but hasn't investigated why
tokens are expiring at ~30 minutes when they're configured for 60 minutes.
The agent is treating the symptom (early expiration) by adding refresh
logic, rather than finding the root cause. The fact that tokens consistently
expire at 30-32 minutes suggests either: (1) a configuration mismatch
somewhere else overriding the 3600-second setting, (2) a timezone/clock
issue between server and client, or (3) the JWT library might be using a
different time unit or has a default max age.
Fix-A.1: Stop adding refresh logic and investigate the actual token
expiration time. Add logging to print the exact 'exp' claim value when
tokens are created and when they're validated. Check if there's another
config file, environment variable, or hardcoded value setting token
expiration to 1800 seconds (30 min). Also verify the JWT library's time unit
- some libraries use milliseconds while others use seconds. The issue is
likely a simple configuration problem, not a need for complex refresh
mechanisms.
Fix-A.2: The solution the agent is trying to implement is overly complex,
confusing, and not tackeling the root cause. The agent should take a step
back and think about what it is actually trying to do, discard its current
approach and come up with a simpler solution that is more likely to work.
It should recall SE best practices and clean code principles.

Problem-B: Agent has successfully modified token validation and refresh logic,
but the integration test continues to fail at exactly 32 minutes. Despite the
previous guidance to investigate configuration mismatches, the agent hasn't
checked for environmental differences between unit tests (which pass) and
integration tests (which fail). The agent also hasn't noticed that two
different JWT libraries are imported (PyJWT and python-jose), which could
mean tokens are created with one library but validated with another.
Additionally, the agent keeps focusing on server-side fixes without considering
that the test client might have its own timeout or token handling logic
that's causing the consistent 32-minute failure.
Fix-B.1: Audit which JWT library is actually being used where. Search for
`from jose import` and `from jwt import` patterns across the codebase. Create
a simple debug endpoint that generates a token and immediately decodes it
with both libraries to see if they interpret expiration differently. The
symptom of tokens expiring at ~32 minutes (close to but not exactly 30)
could indicate timestamp precision or timezone handling differences between
the libraries. Consider standardizing on one JWT library throughout the codebase.
</REFLECTIONS>
```

Figure 13: Part 2 of our joint critic and summarization LLM-Summary prompt. Compared to the LLM-Summary prompt we use in our main experiments (Figure 11), we also prompt the LLM to act as execution-free critic regarding the turns it is summarizing.

Joint Critic-Summary Prompt (Part 3)

```
<PREVIOUS_SUMMARY>
...
</PREVIOUS_SUMMARY>
<TURN-0>
...
</TURN-0>
...
<TURN-20>
...
</TURN-20>
```

Figure 14: Part 3 of our joint critic and summarization LLM-Summary prompt. Compared to the LLM-Summary prompt we use in our main experiments (Figure 11), we also prompt the LLM to act as execution-free critic regarding the turns it is summarizing.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction are discussed in detail in Section 4 and Section 5. Additionally, we extensively document our experimental setup in Section 3 and take steps to ensure our experiments cover diverse configurations and find that our experimental results support our claims across all configurations.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We explicitly document the limitations of our work and the threats to the validity of our study in Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: Our work presents an extensive, empiric study of the cost-performance tradeoff of SE agent context management and thus does not contain theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: In Section 3 we comprehensively document our experimental setup, including the models used, their hyperparameters, and the benchmark we evaluate on. Furthermore, we provide details on our chosen configurations in Appendix D.4 and Appendix D.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We release our LLM-summary implementation[9] for SWE-agent [35]. The experimental results of our main experiments can be openly accessed via HuggingFace[10].

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: In Section 3 we comprehensively document our experimental setup, including the models used, their hyperparameters, and the benchmark we evaluate on. Furthermore, we provide details on our chosen configurations in Appendix D.4 and Appendix D.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: In Section 3 we show the standard deviation of the solve rate, cost and context window size in our preliminary experiments.

[9] `https://github.com/JetBrains-Research/the-complexity-trap`

[10] `https://huggingface.co/datasets/JetBrains-Research/the-complexity-trap`

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We present details on our infrastructure in Section 3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our findings pave the way toward effective and efficient LLM agents, offering an immediate way of reducing the impact of AI on our climate and environment.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We highlight the positive impact of reduced computational costs on our environment in Section 7.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our work presents an extensive evaluation regarding the efficiency of existing SE agent systems, using existing models and an existing benchmark and thus does not pose any such risk.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We credit the authors of SWE-bench [11], SWE-bench Lite-50 [3], SWE-bench Verified [5], SWE-agent [35] and OpenHands [30] at the appropriate locations in our work in Section 2, Section 3, and Section 4.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release our implementation for generating LLM-summaries in SWE-agent [35] with this work in a well-documented manner.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: All of our experiments are conducted on existing benchmarks. Our work does not involve any crowdsourced labor.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: All of our experiments are conducted on existing benchmarks. Our work does not involve any crowdsourced labor.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: Our work focuses on the efficiency on AI agents in SE, thus we explicitly discuss the use of specific LLMs throughout our work, for example in Section 3 and Section 4.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.