Simple, Scalable Reasoning via Iterated Summarization

Anonymous Authors[1]

Abstract

Training large language models to “think” longer by generating chains of thought has led to breakthroughs in their reasoning capabilities. However, their limited context length is a barrier to scaling their thinking time even further. We investigate iterated summarization as a practical approach to extend thinking time: models alternate between summarizing lengthy reasoning traces and reasoning about the problem given summaries of previous attempts. There are many possible summarization strategies, so a key scientific question emerges: what types of summaries effectively compress lengthy reasoning traces? To study this, we investigate the design space of summarization strategies and evaluate their performance in the context of iterated summarization. On AIME 2024 & 2025, our best iterated summarization method, which preserves backtracking behavior, boosts accuracy by 11% over initial reasoning attempts and significantly surpasses baseline methods of extending test-time compute.

1. Introduction

Optimizing language models for reasoning via reinforcement learning has led to breakthrough advances (OpenAI, 2024; DeepSeek-AI et al., 2025). These reasoning models show dramatic improvements over their non-reasoning counterparts on tasks such as math and coding. Much of this improvement comes from “thinking” longer, which has motivated test-time techniques to extend how long they think (Muennighoff et al., 2025).

Although these techniques reliably improve performance, they all increase the length of reasoning traces, leading to a substantial compute and memory overhead. Various approaches have been explored for reducing the computational cost of reasoning. For example, some approaches use supervised finetuning (Yan et al., 2025) or reinforcement learning to teach the model to generate shorter reasoning traces (Aggarwal & Welleck, 2025). Other approaches compress reasoning into special tokens (Xia et al., 2025; Cheng & Durme, 2024). While these approaches can be effective, they require specialized training techniques. The complexity of these methods motivates a natural question: are there simple yet effective test-time interventions that can be applied without additional training to manage continuously expanding context windows?

Given that the growing context window is the key bottleneck to extending reasoning, a natural approach is to periodically summarize the reasoning trace. In this work, we investigate *iterated summarization* (IS) as a practical framework for scalably extending thinking, where models alternate between generating reasoning traces and summarizing these traces to inform their future reasoning (Figure 1). IS is a training-free, test-time intervention that leverages the strong summarization capabilities of pretrained LLMs.

Although summarization is an intuitive approach, the specific method of summarization is clearly a critical design choice. While properties of effective summaries are well-established in standard domains like question answering, the properties of effective summaries of *reasoning* are less clear.

How do we compress reasoning traces through summaries in a way that facilitates improved reasoning? What elements are crucial to preserve—intermediate calculations, or only the path to the final answer? To address these questions, we systematically explore a range of summarization strategies, from simple heuristics that extract the most recent tokens or final answer to more sophisticated approaches using LLMs. Our investigation reveals that the most effective method uses an LLM to summarize reasoning traces with emphasis on moments of backtracking during the reasoning trace. This approach delivers an 11% average improvement over initial reasoning attempts and outperforms baseline methods of extending test-time compute.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Under review by the Workshop on Long-Context Foundation Models (LCFM) at ICML 2025. Do not distribute.

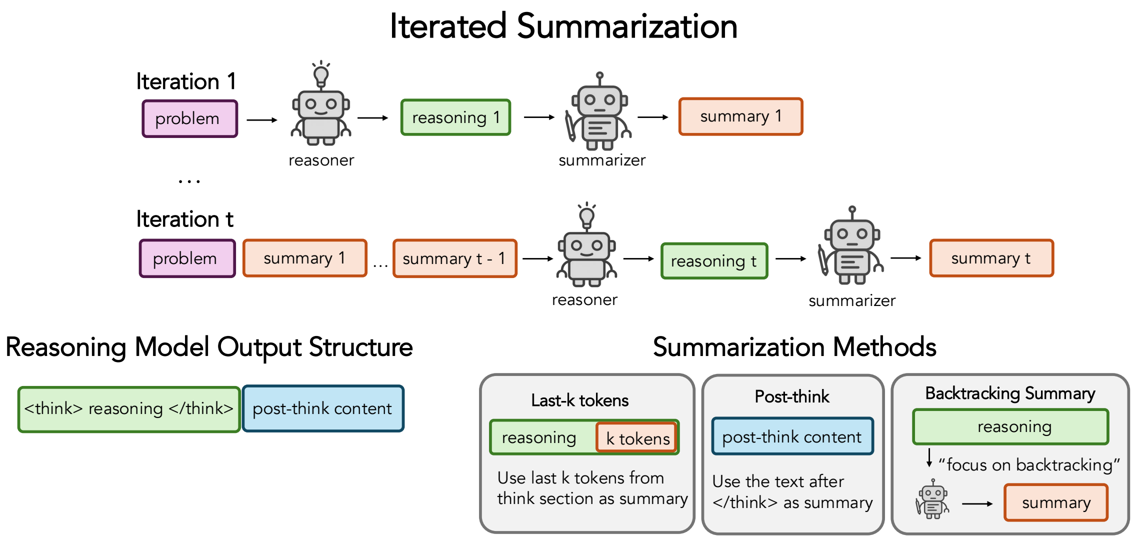

Figure 1. Iterated Summarization Overview. **Top:** Iterated summarization alternates between summarizing lengthy reasoning traces and using those summaries as context for a reasoning model's next attempt at solving a problem. **Left:** structure of reasoning model outputs, divided into reasoning and post-think content. **Right:** Illustration of three summarization methods: Last-k, Post-Think, and Backtracking summary.

In summary, our contributions are as follows:

- We introduce **Iterated Summarization** (IS), a simple but effective technique that alternates between generating reasoning traces and summarizing them, enabling language models to "think longer" without drastically increasing the context window.
- We explore multiple summarization techniques for compressing reasoning traces and identify behaviors that unlock greater reasoning capabilities.
- On AIME 2024 and AIME 2025, our best IS technique delivers an 11% performance boost.

2. Method

Our goal is to understand the characteristics of effective summarization strategies that enable an LLM to reason longer while keeping the context length manageable. This way, we can benefit from longer reasoning without paying a prohibitive computational and memory cost. This section introduces the iterated summarization (IS) framework.

2.1. Iterated Summarization

At a high level, IS alternates between reasoning and summarization, as illustrated in the top portion of Figure 1. In iteration t, a **reasoning model** R attempts to solve a problem by producing a reasoning trace r_t. We then pass r_t to a **summarizer** S that takes r_t as input and produces a shorter summary s_t (see 2.2 for summarization methods). We regenerate a solution to the problem by prompting the reasoning model *R* with all prior summaries $s_1 ... s_{t-1}$ included after the question, in chronological order. We repeat this procedure for T iterations. We fix T=5 in our experiments, for a total of five reasoning iterations and four summarization steps.

2.2. Summarization Methods

In this section, we motivate and introduce the different summarizers that we compare.

Base Summary Our base summarization method simply prompts an LLM to summarize the reasoning trace. This method allows us to understand what the summarizer preserves without more specific instructions.

Backtracking Summary Recent work on characterizing the reasoning strategies of LLMs (Gandhi et al., 2025; Venhoff et al., 2025) has highlighted backtracking–where the reasoner changes or revises the approach–as a key "cognitive behavior" in successful reasoning. Motivated by this, we designed a backtracking summarization prompt that instructs the summarizer to highlight moments where the reasoning model changed its approach or revised its thinking. This was inspired by the insight that highlighting these instances of backtracking in summaries may encourage subsequent reasoning iterations to adopt more effective problem solving approaches.

Our next few approaches do not use an LLM based summarizer and instead use simple functions of the reasoning

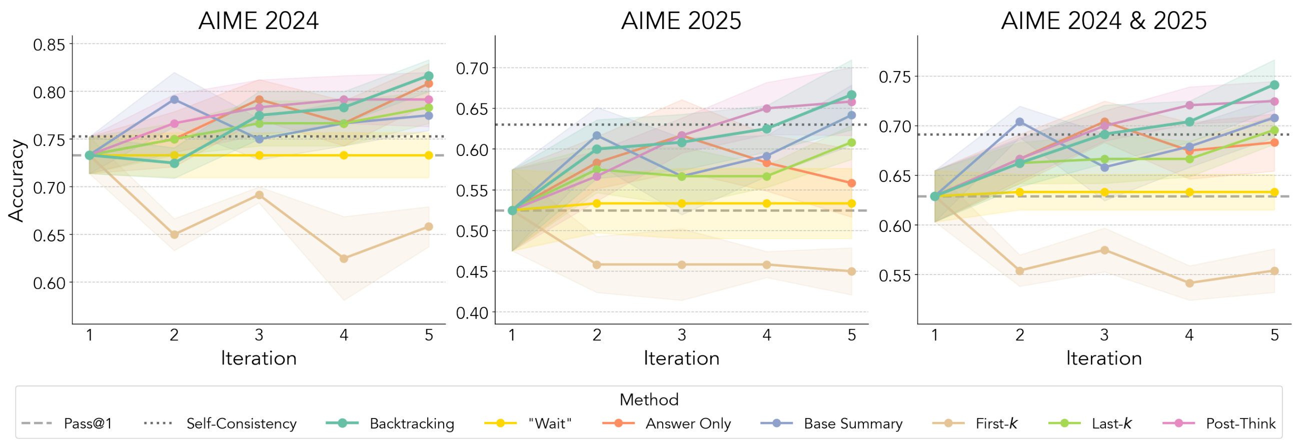

Figure 2. Accuracy on AIME 2024 and 2025 problems by iteration for different methods of extending test-time compute. Shaded regions represent $\pm$ 1 SEM.

model's output. To understand them, it is helpful to understand how open-weights reasoning models structure their responses (shown in the bottom left of Figure 1): they typically produce a <think> tag to indicate the start of their reasoning process, terminate their reasoning process with a closing </think> tag, and then provide the "post-think" answer: a clean response that describes their final answer in some detail.

Post-Think We extract this "post-think" answer by parsing the tokens after the </think> and use it as the summary. This text is often a brief solution justifying the final answer, which is a natural summary of the final approach.

Last-k truncation Within a reasoning trace, later reasoning steps may supersede or invalidate initial steps. We operationalize this insight by choosing the last k tokens before the </think> tag, capturing the most recent steps. To control for token length, we set $k = 404$ for Last-k (and First-k, mentioned later), matching the average length of Base summaries on the combined AIME dataset.

2.3. Baselines

Self-consistency We use self-consistency as a baseline (Wang et al., 2023) which samples multiple solutions from the model and takes a majority vote over the answers. For fair comparison with IS, we sample the same number of solutions as iterations in our IS methods ($T = 5$).

Budget forcing ("Wait") Following Muennighoff et al. (2025), we extend test-time thinking by forcing R to extend its reasoning by replacing the closing </think> tag with "Wait", forcing the model to continue reasoning. Unlike IS, budget forcing is limited by the context length since the full reasoning trace is kept in context.

First-k truncation As a negative control, we also consider first-k truncation and choose the first k tokens after the opening <think> tag. These early tokens primarily contain only problem setup rather than reasoning or critical insights.

Answer Only As an ablation, we explore the effectiveness of only providing the final answer from previous attempts, without any justification. We use the statement "The answer is {`final answer`}" as the summary.

3. Results & Analysis

3.1. Experimental Setup

Models that achieve near ceiling performance have less room to benefit from increasing thinking time. Therefore, in all experiments, we use the `DeepSeek-R1-Distill-Qwen-14B` model as our reasoner R and the `Qwen2.5-14B-Instruct` model as our summarizer S; the `DeepSeek-R1-Distill-Qwen-14B` model was trained by finetuning the `Qwen2.5-14B-Instruct` model on `Deepseek-R1` reasoning traces. We also experimented with using `DeepSeek-R1-Distill-Qwen-14B` itself as the summarizer but found that it would often attempt to solve the problem instead of summarizing.

We evaluate on AIME 2024 and AIME 2025, consisting of 60 high-school mathematics competition problems in English (Mathematical Association of America, 2024; 2025). Each solution was generated with temperature 0.6 and top-p 0.95, matching the sampling parameters used in the original DeepSeek R1 paper (DeepSeek-AI et al., 2025), with a maximum generation length of 32,768 tokens. Experiments

Method	**AIME 2024**	**AIME 2025**	**Combined**
Pass@1	0.733 ± 0.019	0.525 ± 0.050	0.629 ± 0.026
Self-Consistency	0.753 ± 0.056	0.630 ± 0.059	0.692 ± 0.055
"Wait"	0.733 ± 0.024	0.533 ± 0.043	0.633 ± 0.018
Answer Only	<u>0.808</u> ± 0.021	0.558 ± 0.042	0.683 ± 0.029
Post-Think	0.792 ± 0.028	<u>0.658</u> ± 0.042	<u>0.725</u> ± 0.020
First-k	0.658 ± 0.021	0.450 ± 0.029	0.554 ± 0.022
Last-k	0.783 ± 0.017	0.608 ± 0.021	0.696 ± 0.014
Base Summary	0.775 ± 0.016	0.642 ± 0.037	0.708 ± 0.020
Backtracking Summary	**0.817** ± 0.017	**0.667** ± 0.043	**0.742** ± 0.025

Table 1. Accuracy (± 1 SEM) across methods for AIME 2024, AIME 2025, and combined datasets. The best method is bold and the second-best is underlined.

Method	**Improved (%)**	**Regressed (%)**
	wrong$_{t=1}$ → correct$_{t=5}$	correct$_{t=1}$ → wrong$_{t=5}$
"Wait"	2.00 ± 2.00	**0.60** ± 0.60
Answer-Only	<u>30.26</u> ± 6.76	8.86 ± 3.55
Post-Think	30.11 ± 4.00	2.57 ± 1.86
First-k	21.85 ± 3.64	24.99 ± 3.70
Last-k	23.19 ± 2.71	3.23 ± 2.50
Base Summary	27.88 ± 2.90	4.03 ± 0.85
Backtracking Summary	**31.81** ± 3.49	<u>0.66</u> ± 0.66

Table 2. Stability of methods for extending test-time compute (± 1 SEM) The best method is bold and the second-best is underlined.

were run on an A100-80GB node hosted by Fireworks AI. Full prompts, hyperparameters, and experiment details are provided in the Appendix. To account for decoding stochasticity, we run every experimental condition four independent times. We report the mean accuracy and the standard error of the mean (± SEM) across the runs.

3.2. Iterated Summarization Boosts Reasoning

Table 1 shows that all of our IS methods (excluding our negative control, First-k) outperform Pass@1 on both AIME 2024 & 2025. We bold the best method and underline the second-best. Our best method, Backtracking Summary, delivers significant accuracy gains of +8.4% on AIME 2024, +14.2% on AIME 2025, and **+11.3%** overall compared to Pass@1. Comparing to Self-Consistency, our Backtracking Summary method maintains an advantage with improvements of +6.4% on AIME 2024, +3.7% on AIME 2025, and +5.0% on the combined dataset. Figure 2 plots accuracy *per iteration*, with Backtracking Summary and Post-Think increasing nearly monotonically on average.

The "Wait" baseline improves over Pass@1 by a small +0.4% margin on the combined dataset; our Backtracking method has a **+10.9%** improvement over "Wait". Each reasoning continuation after the "Wait" token averages only 1,345 tokens, whereas each new reasoning attempt for Backtracking summarization generates 9,018 tokens on average (see Appendix Figure 7 for full token count distributions). This highlights a fundamental difference between how these approaches extend test-time thinking: budget forcing only extends previous reasoning where it left off, while IS allows the model to re-attempt full solutions. Interestingly, even the Answer Only method (67.1% accuracy) outperforms the "Wait" baseline, demonstrating that even minimal information from prior attempts can be valuable for subsequent reasoning attempts.

We examined reasoning traces and their summaries to gain insights into the properties of Iterated Summarization. Unlike methods like self-consistency that generate independent solutions, IS enables the model to retain relevant portions of the previous attempts and focus effort on trying something new. This progressive improvement is evident when early iterations establish correct foundations (such as setting up equations or coordinate systems) but make errors in later steps. Subsequent iterations often preserve these foundations, spending compute on exploring new approaches. In contrast to other summarization methods, backtracking summaries often describe abandoned approaches, allowing the model to learn from these attempts.

3.3. Stability of Iterated Summarization

We also evaluate two additional key properties: whether summarization enables us to solve problems in later iterations that were unsolved in the initial reasoning attempt (**improvement**), while minimizing cases where solved problems become incorrect later (**regression**).

We consider two sets of problems: ones that the model gets correct at iteration 0 and ones that the model gets incorrect at iteration 0. We then revisit these same problems after applying our IS method, and characterize how many problems improve (wrong$_{t=1}$ → correct$_{t=5}$) vs. regress (correct$_{t=1}$ → wrong$_{t=5}$). Empirically, our best IS method (backtracking) yields the highest Improvement-to-Regression ratio. On problems that an initial reasoning attempt does not solve, IS with backtracking summaries solves **31.81%** of these problems after iteration 5. On problems that an initial reasoning attempt gets right, only **0.66**% of these problems deteriorate to an incorrect answer after iteration 5. This suggests that IS is a **stable** method of increasing inference-time compute.

Conclusion

Iterated Summarization is a framework that alternates between reasoning and summarization of reasoning traces to extend a model's thinking time while managing the challenges that come with longer reasoning. On the AIME 2024 & 2025 benchmarks, our best variant, Backtracking Summary, boosts accuracy by over 11% compared to Pass@1, while also outperforming self-consistency and "Wait" baselines. Crucially, these gains are stable: later iterations correct 31.81% of previously unsolved problems while regressing on only 0.66% of solved ones.

Impact Statement

This paper presents IS, a lightweight, model-agnostic framework that empirically extends the reasoning capabilities of LLMs at inference-time. This method can be used in a variety of applications that require a boost in performance without additional post-training or compute requirements. While it is possible that powerful reasoning models could present certain risks when applied to sensitive domains, our proposed framework is unlikely to introduce or magnify these risks. IS enhances model reasoning through efficient context management and iterative thinking, allowing models to operate within existing safety frameworks and constraints.

References

Aggarwal, P. and Welleck, S. L1: Controlling how long a reasoning model thinks with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.04697.

Cheng, J. and Durme, B. V. Compressed chain of thought: Efficient reasoning through dense representations, 2024.

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Ding, H., Xin, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Wang, J., Chen, J., Yuan, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Ye, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Zhao, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Xu, Y., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., and Zhang, Z. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Gandhi, K., Chakravarthy, A., Singh, A., Lile, N., and Goodman, N. D. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars, 2025. URL https://arxiv.org/abs/2503.01307.

Mathematical Association of America. Maa invitational competitions - American Invitational Mathematics Examination. Mathematical Association of America, February 2024. URL https://maa.org/maa-invitational-competitions/. Competition examination.

Mathematical Association of America. Maa invitational competitions - American Invitational Mathematics Examination. Mathematical Association of America, February 2025. URL https://maa.org/maa-invitational-competitions/. Competition examination.

Muennighoff, N., Yang, Z., Shi, W., Li, X. L., Fei-Fei, L., Hajishirzi, H., Zettlemoyer, L., Liang, P., Candès, E., and Hashimoto, T. s1: Simple test-time scaling, 2025. URL https://arxiv.org/abs/2501.19393.

OpenAI. Openai o1 system card. Technical report, OpenAI, September 2024.

Venhoff, C., Arcuschin, I., Torr, P., Conmy, A., and Nanda, N. Understanding reasoning in thinking language models via steering vectors. In *Workshop on Reasoning and Planning for Large Language Models*, 2025. URL https://openreview.net/forum?id=OwhVWNOBcz.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. Huggingface's transformers: State-of-the-art natural language processing, 2020. URL https://arxiv.org/abs/1910.03771.

Xia, H., Li, Y., Leong, C. T., Wang, W., and Li, W. Tokenskip: Controllable chain-of-thought compression in llms, 2025.

Yan, Y., Shen, Y., Liu, Y., Jiang, J., Zhang, M., Shao, J., and Zhuang, Y. Inftythink: Breaking the length limits of long-context reasoning in large language models. *arXiv preprint arXiv:2503.06692*, 2025.

A. Appendix

A.1. Artifact License Details

- **Model weights.** We rely on two open-weight LLMs: `DeepSeek-R1-Distill-Qwen-14B`[1] and `Qwen2.5-14B-Instruct`[2], both released under the MIT license.
- **Benchmark data.** The AIME 2024 & 2025 problem sets are in the public domain (problems reproduced from the Art of Problem Solving archive).
- **Code and prompts.** Our implementation, prompts, and evaluation scripts will be released on GitHub under the permissive MIT license.

A.2. Hyperparameter & Experiment Details

Both DeepSeek-R1-Distill-Qwen-14B and Qwen-2.5-14B-Instruct have 14.7 billion parameters each. Running the main experiments (4 seeds × 60 problems × 5 iterations) consumed approximately 45 GPU-hours on a single NVIDIA A100-80GB.

We use Hugging Face models and tokenizers for running models and tokenization (Wolf et al., 2020).

We use these sampling parameters for experiments:

Parameter	Reasoning Model	Summarizer
max_tokens	32768	32768
temperature	0.6	0.6
top_p	0.95	0.95
top_k	40	40
presence_penalty	0	0
frequency_penalty	0	0

Table 3. Sampling parameters for reasoning and summarization models.

These parameters were chosen to maintain consistency with the original DeepSeek-R1 paper (DeepSeek-AI et al., 2025). The max_tokens value was set high enough to accommodate the longest reasoning traces while avoiding truncation.

Algorithm 1 Iterated Summarization (IS)

```
Require: question $q$, reasoning model $R$, summarizer $S$, iterations $T$
1: $\Sigma \leftarrow [\,]$                                   ▷ list of summaries
2: for $t = 1$ to $T$ do
3:     $r_t \leftarrow R\big(q,\ \texttt{summaries} = \Sigma\big)$
4:     if $t < T$ then
5:         $s_t \leftarrow S(r_t)$                            ▷ compress trace
6:         $\Sigma$.append$(s_t)$
7:     end if
8: end for
9: return final answer extracted from $r_T$
```

A.3. Additional Figures

Approach	Iter 1	Iter 2	Iter 3	Iter 4	Average
Backtracking	2.14	2.09	2.00	1.94	2.04
Base Summarization	0.45	0.44	0.39	0.50	0.45

Table 4. Average Backtracking Behavior Counts For Summaries Across Iterations

[1] `https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-14B`
[2] `https://huggingface.co/Qwen/Qwen2.5-14B-Instruct`

Method	Iter 1	Iter 2	Iter 3	Iter 4	Iter 5	Overall
Backtracking	1.28	1.79	2.00	1.88	1.98	1.78
Base Sum	1.46	2.00	2.14	1.87	2.16	1.93

Table 5. Average Backtracking Behavior Counts For Reasoning Across Iterations

AIME Problems Prompt

Solve the following AIME problem. All answers are integers ranging from 0 to 999, inclusive. Report your answer in `\boxed{}` format.

PROBLEM:
{question}

Figure 3. AIME Problem Prompt Template

Base Summarization Prompt

Summarize the following attempted solution to the problem:

PROBLEM:
{`question`}
ATTEMPTED SOLUTION:
{`reasoning`}
SUMMARY:

Figure 4. Base Summarization Prompt Template

Backtracking Summarization Prompt

Summarize the following attempted solution to the problem, emphasizing the instances where the model changed its strategy, revised a previous decision, or explicitly backtracked from a prior line of reasoning.

PROBLEM:
{`question`}
ATTEMPTED SOLUTION:
{`reasoning`}
SUMMARY:

Figure 5. Backtracking Summarization Prompt Template

Improve Using Summaries Prompt

Solve the following AIME problem. All answers are integers ranging from 0 to 999, inclusive. Report your answer in `\boxed{}` format.

PROBLEM:
`{question}`
Here are summaries of your previous solution attempts:
`{summaries}`

Based on your previous solution attempts, evaluate whether the most recent approach and answer are correct. If not, consider a different approach.

Figure 6. Prompt Template for later iterations to use and build from previous summaries

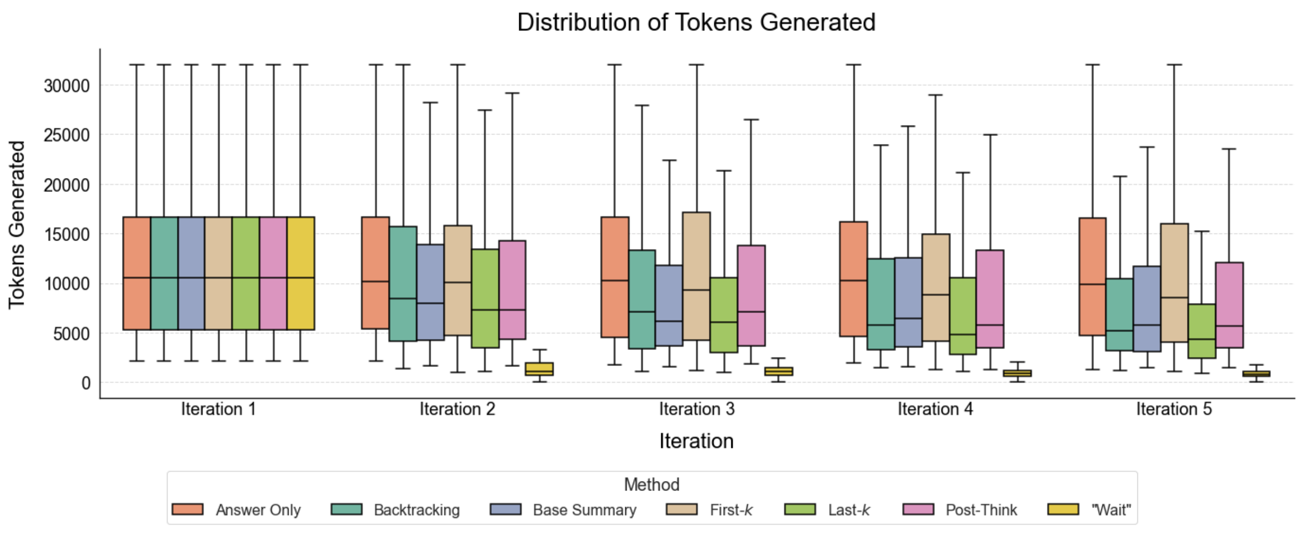

Figure 7. Token Count Distribution

Method	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5
Answer Only	12438.47 ± 521.66	11839.93 ± 488.61	11821.38 ± 517.14	11672.89 ± 503.48	11939.12 ± 535.31
Backtracking	12438.47 ± 521.66	10609.80 ± 509.23	9036.55 ± 424.25	8444.58 ± 425.09	7979.73 ± 438.79
Base Summary	12438.47 ± 521.66	9596.15 ± 401.03	8427.26 ± 393.72	9116.83 ± 484.48	8246.03 ± 421.15
First-k	12438.47 ± 521.66	11517.05 ± 514.59	11451.91 ± 543.41	10740.58 ± 496.92	10849.08 ± 522.41
Last-k	12438.47 ± 521.66	9225.58 ± 439.45	7627.92 ± 372.73	7438.36 ± 417.94	5919.89 ± 320.07
Post-Think	12438.47 ± 521.66	9759.13 ± 476.39	9240.95 ± 419.28	8854.25 ± 465.18	8341.21 ± 419.39
“Wait”	12438.47 ± 521.66	1635.50 ± 104.94	1390.48 ± 92.66	1243.05 ± 97.79	1109.97 ± 83.68

Table 6. Summary statistics (± SEM) for each method across iterations.