Increasing the Thinking Budget is Not All You Need

Anonymous ACL submission

Abstract

Recently, a new wave of thinking-capable Large Language Models has emerged, demonstrating exceptional capabilities across a wide range of reasoning benchmarks. Early studies have begun to explore how the amount of compute in terms of the length of the reasoning process, the so-called *thinking budget*, impacts model performance. In this work, we propose a systematic investigation of the thinking budget as a key parameter, examining its interaction with various configurations such as self-consistency, reflection, and others. Our goal is to provide an informative, balanced comparison framework that considers both performance outcomes and computational cost. Among our findings, we discovered that simply increasing the thinking budget is not the most effective use of compute. More accurate responses can instead be achieved through alternative configurations, such as self-consistency and self-reflection.

1 Introduction

Large language models (LLMs) have made remarkable strides in recent years, achieving unprecedented levels of performance in complex reasoning tasks. Recent efforts have been designed not only to produce answers, but also to show an explicit step-by-step reasoning process that leads to the answer (OpenAI, 2024; Anthropic, 2025; Google DeepMind, 2025; DeepSeek-AI, 2025; Qwen Team, 2025; Meta-AI, 2025) .

Some works have been exploring how these models behave when their *thinking budget*–the maximum amount of thinking tokens– is taken into account and its relation with model's performance. Muennighoff et al. (2025) for instance investigated the scaling, showing that increasing compute during inference improves overall performance of the models. However, little is known about how these reasoning models perform in more *agentic* configurations. On the other hand, Wang et al. (2024) perform a thoughtful analysis comparing different reasoning approaches such as single agent, multi-agent, reflection, self-consistency, etc, from the perspective of the reasoning budget, but without dealing with explicit reasoning models. Nevertheless, there has been no effort yet to understand how the amount of compute performs depending on the configuration where it is used. In this work, we design a complete suite of configurations that, together with the compute budget, provide a balanced perspective on reasoning strategy effectiveness. These configurations vary across multiple dimensions, including model size, inference-time optimization techniques, allowing us to isolate and examine the role that compute plays in different operational settings. By systematically evaluating these variables, we aim to uncover nuanced interactions between compute allocation and reasoning performance, ultimately guiding more efficient and adaptive deployment strategies for intelligent systems.

We compare several established reasoning strategies, such as Self-Consistency and Reflection, and explore how these methods perform when combined with varying levels of thinking budgets on challenging benchmarks. By systematically varying the amount of compute allocated to each strategy, we aim to understand how resource distribution influences reasoning quality and decision-making accuracy. Our empirical results unveil compelling insights into how and where to assign computational budgets to maximize overall performance, highlighting trade-offs and synergies across different strategies and tasks.

Our contributions are threefold:

- We introduce a comprehensive framework that evaluates reasoning strategies under diverse compute configurations, providing a structured view of performance trade-offs.
- We present extensive empirical results on a

set of challenging benchmarks, identifying optimal budget allocation patterns across reasoning techniques. We will show that straightforward extending the thinking budget might not be the best strategy.

- We offer actionable guidelines for effectively combining compute and reasoning strategies to achieve better outcomes in real-world AI applications.

2 Past work

Since the introduction of GPT-3 (Brown et al., 2020), many efforts have been made to improve the capabilities of the models by applying different prompting techniques. Wei et al. (2022) introduced *Chain of thought* (CoT), a technique that later became the de facto initiation technique of LLMs. Empirical results show that CoT improves reasoning capability by encouraging models to plan step-by-step and reason toward the answer, using computation to decompose tasks into smaller and simpler steps. Later, Taylor et al. (2022) introduced Galactica, an LLM specifically trained to be used in the science domain. Their training data included not only scientific text, but also LaTeX code from scientific publications, programming code, DNA sequences, etc. Galactica also introduced the idea of having an explicit portion of the generation for step-by-step reasoning, called working memory context. This generation portion was surrounded by a special token <*work*>. This idea was further exploited by OpenAI (2024) with the introduction of o1-preview, a model designed to *spend more time thinking* before responding. However, this model and the successors do not make the thinking process public. Recently, DeepSeek-AI (2025) released the DeepSeek-R1 family of models, an open-source family trained specifically to reward answers that format its thinking process between <*think*> and <*/think*> tags.

2.1 Analyzing the compute budget

Su et al. (2025) performed a thoughtful analysis of the relationship between reasoning length and answer correctness. This work shows that the length of LLM reasoning is not a good indicator of perceived difficulty, failing to increase appropriately for hard problems. This work also finds out that LLMs often misjudge problem difficulty and shows that optimizing for shorter responses can reduce length while maintaining reasonable accuracy. Similarly, the analysis performed by Wang et al. (2024) is also an analysis paper that does not deal with reasoning LLMs but compares different reasoning approaches, single agent, multi-agent, reflection, self-consistency, etc, from the perspective of the reasoning budget. A key finding is that complex reasoning strategies often do not outperform simpler baselines due to inherent algorithmic superiority, but rather because they are allocated larger computational resources. In a similar path, Snell et al. (2024) explores how increasing the token budget of LLMs inference time can significantly boost their performance. Two strategies are investigated: "dense, process-based verifier reward models" and "adaptively updating the model's distribution over a response.". As an outcome, the authors discovered that a strategy that allocates more computational effort based on prompt difficulty is more efficient.

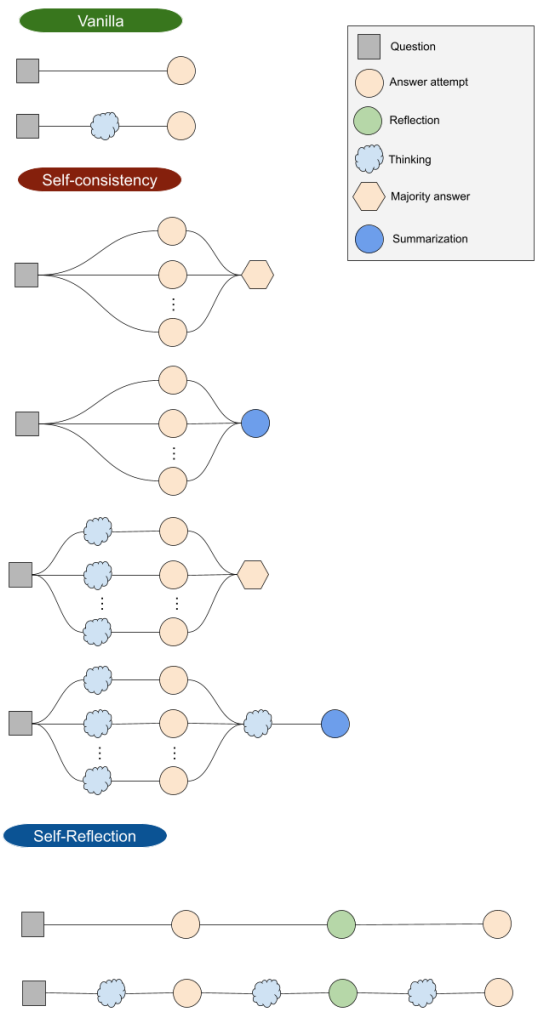

Figure 1: Visualization of reasoning strategies across different configurations.

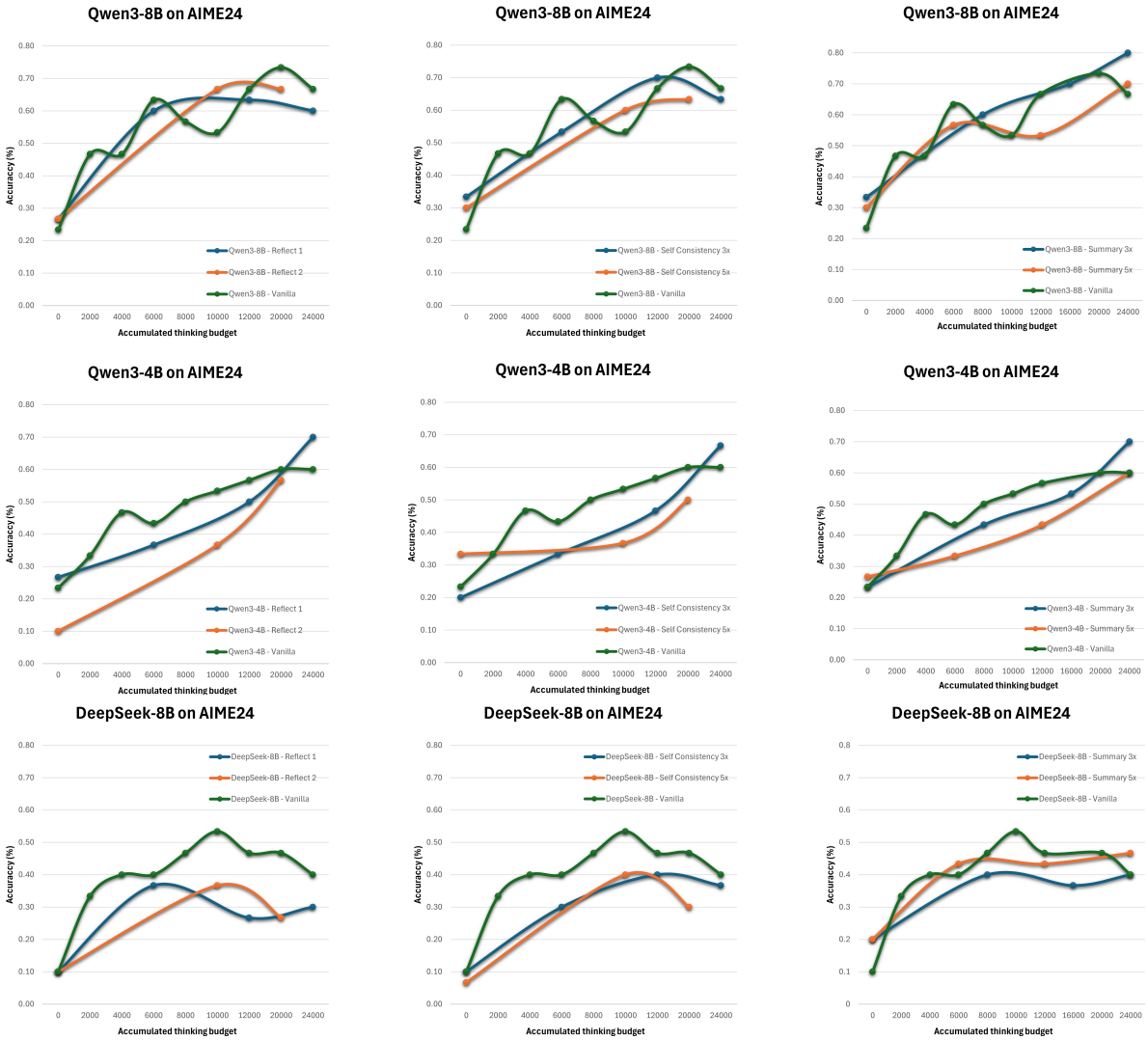

Figure 2: Evaluation of different configurations on the AIME24 on Qwen3-8B, Qwen3-4B and DeepSeek-R1-Distill-Llama-8B.

3 Thinking

Here we define what we understand by compute budget. As empirical results will show, performance often correlates with the number of thinking tokens, regardless of the amount of compute without thinking. From now on we use thinking token (and not total token output) as our metric.

3.1 Reasoning Budget

Total number of thinking tokens. This is the expected number of tokens between special *<think>* tags. In our experiments we do not set this value as a maximum, but we employ a forcing mechanism, following Muennighoff et al. (2025) that guarantees a minimum amount of tokens as defined, by inserting "Wait" tokens if the budget is not met.

Number of API calls. This is the amount of times the LLM is prompted to generate an answer. This value is independent from the thinking budget. For instance when we consider a 6,000 token budget, we analyze configurations where the entire budget is consumed in a single API call, or divided into three calls with 2,000 thinking tokens each. This aspect will be clarified in the Experiments section below.

3.2 Optimizing compute

In this section, we present different configurations aimed at answering this research question. We can visualize them in Figure 1. Each configuration is tested with varying levels of the thinking process, including configurations with no thinking at all:

Vanilla. The model is directly prompted with a question and returns an answer without any additional reasoning steps.

Self-consistency. The model is prompted multiple times with temperature set to 1.0. The final answer is selected based on majority voting among the generated responses. We experimented with 3x and 5x consistency runs.

Summary. Similar to Self-Consistency, but instead of relying on majority voting, an additional call to the LLM is made to summarize and consolidate all attempted answers into a single, unified response. We also experimented with 3x and 5x consistency runs.

Reflection. Inspired by the approach proposed by Madaan et al. (2023). This method involves generating an initial answer and then prompting the model to provide feedback on it. The original answer is subsequently refined on the basis of this feedback. We allowed one and two steps of reflection.

4 Experiments

For our experiments we chose three thinking models: Qwen3-8B recently introduced by Qwen Team (2025) and its smaller version Qwen3-4B; and DeepSeek-R1-Distill-Llama-8B, from DeepSeek-AI (2025). We use the AIME24 datasets as implemented in Muennighoff et al. (2025). In our experiments, we evaluated seven different configurations of reasoning strategies to assess their performance relative to computational cost. We varied the total computational budget between configurations, ranging from zero (no thinking) to 24,000 reasoning tokens. In Figure 2, we compare model performance across the configurations described above as a function of the compute budget. We compare the Vanilla configuration individually against Reflect, Self-Consistency and Summary on the three selected models. Qwen3-8B being the strongest reasoning model seems to leverage the most of the thinking budget. Vanilla does fluctuate across the budgets, and show some signs of plateau across the board with larger allowances. This can be also seen on Figure 4 located in the Appendix. The other two weaker models, Qwen3-4B and DeepSeek-8B, struggle to fully benefit from the configurations lagging behind on most experiments against Vanilla. The Self-Consistency strategy struggle to provide any improvement, specially on those two models. This may be due to the extraction process. On this condiguration, individual results are extracted via regular expressions and counted. Formatting issues might be problematic in this step, while not being an issue in the remaining configurations. Finally we refer to the Summary strategy, which seems to be the strongest configuration overall, clearly surpassing the performance of Vanilla on Qwen3-8B and providing competitive results on the other two models.

Configuration	Qwen3-8B	Qwen3-4B	DeepSeek-8B
Vanilla	23.33%	23.33%	10.00%
Self Consistency 3x	33.33%	20.00%	10.00%
Reflect 1	26.67%	26.67%	10.00%
Summary 3x	33.33%	23.33%	20.00%
Self Consistency 5x	30.00%	33.33%	6.67%
Reflect 2	26.67%	10.00%	10.00%
Summary 5x	30.00%	26.67%	20.00%

Table 1: Performance when thinking is deactivated.

4.1 Zeroing the thinking budget

Table 1 presents empirical results of experiments conducted with thinking explicitly deactivated, that is, the thinking budget was set to zero across all configurations. Under this constraint, each configuration was limited to a fixed maximum number of tokens, ensuring a fair comparison without the influence of variable reasoning depth. Consistent with findings from earlier experiments, performing two reflection steps does not yield additional benefits over a single reflection. The same applies to Self-Consistency and Summary. Increasing the amount of parallel traces is not bringing further benefits. Among all tested configurations, Summary emerges as the top performer. It enables improved results even in the weakest model of the three. Despite the absence of an extended reasoning phase, this setup still benefits from generating multiple diverse outputs and concluding the most consistent response. This indicates that even modest ensemble strategies can significantly enhance answer quality.

5 Conclusion

In this paper we provide an initial examination of the behavior of several reasoning strategies while varying their thinking budget. We confirmed that increasing the compute budget leads to better performance, but simpler configurations reach a performance ceiling that can be overcome by more sophisticated strategies. Overall the *Summary* strategy seems to be the best-performing.

Limitations

Although our analysis shows interesting findings, we carried out experiments on a limited set of datasets and LLMs. A larger sample both in terms of models and benchmarks is necessary to fully validate our current findings.

AI Assistants

The OpenAI service of ChatGPT was used to polishing authors writing.

References

Anthropic. 2025. Claude 3.7 sonnet. https://www.anthropic.com/news/claude-3-7-sonnet.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Harrison Chase. 2024. Langchain (2022).

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.

Google DeepMind. 2025. Gemini 2.5. https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. *Preprint*, arXiv:2303.17651.

Meta-AI. 2025. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation. https://ai.meta.com/blog/llama-4-multimodal-intelligence/.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *Preprint*, arXiv:2501.19393.

OpenAI. 2024. Introducing openai o1-preview. https://openai.com/index/introducing-openai-o1-preview/.

Qwen Team. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *Preprint*, arXiv:2408.03314.

Jinyan Su, Jennifer Healey, Preslav Nakov, and Claire Cardie. 2025. Between underthinking and overthinking: An empirical study of reasoning length and correctness in llms. *Preprint*, arXiv:2505.00127.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A large language model for science. *Preprint*, arXiv:2211.09085.

Junlin Wang, Siddhartha Jain, Dejiao Zhang, Baishakhi Ray, Varun Kumar, and Ben Athiwaratkun. 2024. Reasoning in token economies: Budget-aware evaluation of LLM reasoning strategies. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19916–19939, Miami, Florida, USA. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

A Experiment configuration

We followed the framework introduced by Muennighoff et al. (2025), http://github.com/simplescaling/s1, which is a modified version of Language Model Evaluation Harness from EleutherAI (Gao et al., 2024).
Call:

```
OPENAI_API_KEY=<YOUR_OPENAI_KEY>
PROCESSOR=gpt-4o-mini
lm_eval --model vllm \
  --model_args <model-params> \
 --tasks aime24_figures,aime24_nofigures \
  --batch_size auto \
  --apply_chat_template
```

All configurations were implemented using LangChain (Chase, 2024) framework. Temperature on inference was set to zero except on Self-Consistency and Summary, where we use 1.0 instead.

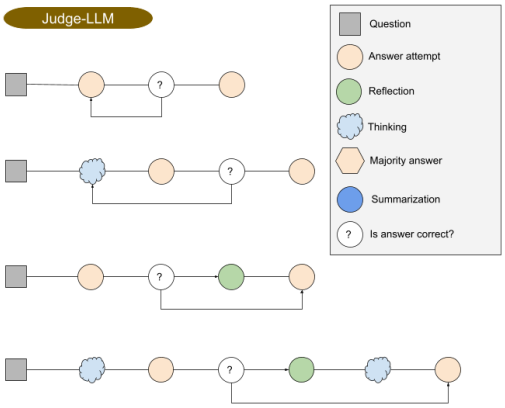

Figure 3: Visualization of reasoning strategies across Judge LLM configurations.

Configuration	Think	Budget	Qwen3-8B
Vanilla		-	23.33%
	✓	2000	46.67%
	✓	4000	46.67%
Judge-LLM w/o history		-	43.33%
	✓	2000	60.00%
	✓	4000	66.67%
Judge-LLM with history		-	46.67%
	✓	2000	63.33%
	✓	4000	63.33%
Judge+Reflection		-	40.00%
	✓	2000	46.67%
	✓	4000	40.00%

Table 2: Comparison between vanilla configuration and Judge LLM.

B Judge LLM

This configuration uses an extra call to the LLM, without thinking to check if the answer provided is correct or another try is needed. Importantly, this judging step does not involve further reasoning; it merely evaluates whether the answer is acceptable or whether a new attempt is required.
Three configurations were considered depicted in Figure 3:

Judge-LLM. After producing an answer, the model is prompted to evaluate whether the response is likely correct. If deemed incorrect, the model is given another opportunity to respond.
Judge-LLM w/history Similar to Judge-LLM, but in this variant, the model's second attempt is informed by both the initial answer and the negative feedback, which are included in the prompt.
Judge+Reflection. A hybrid of the Judge-LLM and Self-Reflection methods. If the initial answer is judged incorrect, the model is prompted to generate feedback and then refine its response accordingly.

Experiments were carried out using Qwen3-8B on AIME24 dataset. Table 2 shows that the inclusion of a Judge LLM appears to be beneficial across a range of configurations and compute budgets. For the first configuration this results is curious since if model values the answer as incorrect, the answer is removed from history, and the model is just answering from scratch. Comparing the first two Judge-LLM configurations, given the mixed results remains unclear whether keeping previous answer is beneficial or not. Meanwhile Judge+Reflection struggles to improve over the Judge-LLM strategy. The result is unclear but an analysis on the results show most of the times the first answer is assess as valid, neglecting the potential improvement from the reflection feedback.

C Prompts used

Here we show the prompts used on the configurations mentioned in the main body of the paper:

Vanilla. and **Self-consistency.**

System

"You are a helpful assistant tasked to answer user's questions"

Summary.

System

"You are a helpful assistant tasked to answer user's questions"

Summarize

Given this question
q
we have a set of potential answers:
$docs$
Take these and distill it into a final, consolidated answer.

Reflection.

System

"You are a helpful assistant tasked to solve puzzles. "
"If the user provides feedback, respond again a revised version of your previous attempts. "
"DO NOT thank for the feedback. Pretend it's the first time you are answering the question. "

Reflect

"You are a helpful assistant able to grade answers. "
"Check correctness of the user's answer. "
"Provide feedback and detailed recommendations, including requests for length, depth, etc.,"

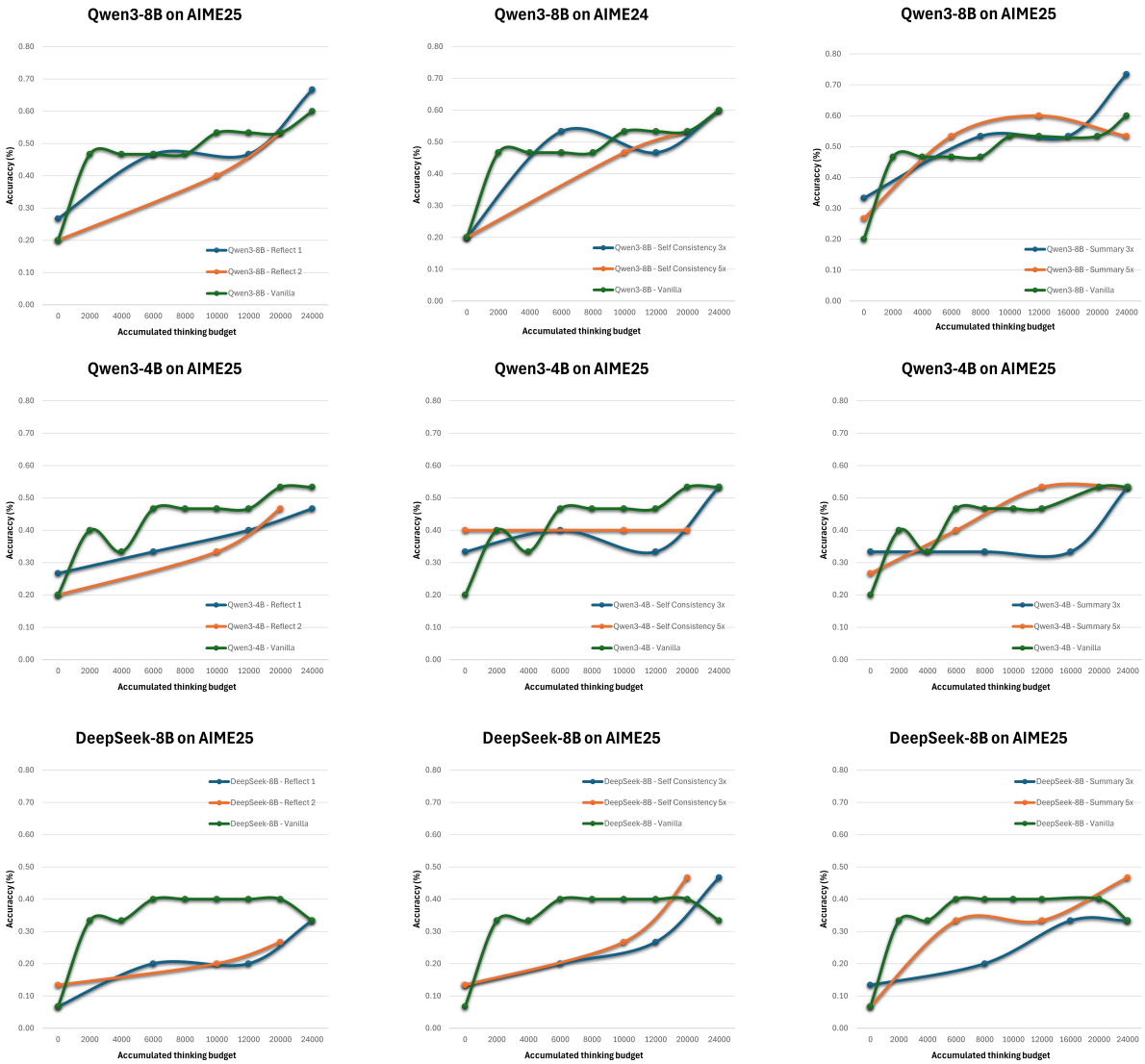

Figure 4: Evaluation of different configurations on the AIME25 on Qwen3-8B, Qwen3-4B and DeepSeek-R1-Distill-Llama-8B.

Configuration	Thinking	Budget	Calls	Total	Qwen3-8B		Qwen3-4B		DeepSeek-8B	
					AIME24	AIME25	AIME24	AIME25	AIME24	AIME25
Vanilla		0	1	0	23.33%	20.00%	23.33%	20.00%	10.00%	6.67%
	✓	2000	1	2000	46.67%	46.67%	33.33%	40.00%	33.33%	33.33%
	✓	4000	1	4000	46.67%	46.67%	46.67%	33.33%	40.00%	33.33%
	✓	6000	1	6000	63.33%	46.67%	43.33%	46.67%	40.00%	40.00%
	✓	8000	1	8000	56.67%	46.67%	50.00%	46.67%	46.67%	40.00%
	✓	10000	1	10000	53.33%	53.33%	53.33%	46.67%	53.33%	40.00%
	✓	12000	1	12000	66.67%	53.33%	56.67%	46.67%	46.67%	40.00%
	✓	20000	1	20000	73.33%	53.33%	60.00%	53.33%	46.67%	40.00%
	✓	24000	1	24000	66.67%	60.00%	60.00%	53.33%	40.00%	33.33%
Self Consistency 3x		0	3	0	33.33%	20.00%	20.00%	33.33%	10.00%	13.33%
	✓	2000	3	6000	53.33%	53.33%	33.33%	40.00%	30.00%	20.00%
	✓	4000	3	12000	70.00%	46.67%	46.67%	33.33%	40.00%	26.67%
	✓	8000	3	24000	63.33%	60.00%	66.67%	53.33%	36.67%	46.67%
Self Consistency 5x		0	5	0	30.00%	20.00%	33.33%	40.00%	6.67%	13.33%
	✓	2000	5	10000	60.00%	46.67%	36.67%	40.00%	40.00%	26.67%
	✓	4000	5	20000	63.33%	53.33%	50.00%	40.00%	30.00%	46.67%
Reflect 1		0	3	0	26.67%	26.67%	26.67%	26.67%	10.00%	6.67%
	✓	2000	3	6000	60.00%	46.67%	36.67%	33.33%	36.67%	20.00%
	✓	4000	3	12000	63.33%	46.67%	50.00%	40.00%	26.67%	20.00%
	✓	8000	3	24000	60.00%	66.67%	70.00%	46.67%	30.00%	33.33%
Reflect 2		0	5	0	26.67%	20.00%	10.00%	20.00%	10.00%	13.33%
	✓	2000	5	10000	66.67%	40.00%	36.67%	33.33%	36.67%	20.00%
	✓	4000	5	20000	66.67%	53.33%	56.67%	46.67%	26.67%	26.67%
Summary 3x		0	4	0	33.33%	33.33%	23.33%	33.33%	20.00%	13.33%
	✓	2000	4	8000	60.00%	53.33%	43.33%	33.33%	40.00%	20.00%
	✓	4000	4	16000	70.00%	53.33%	53.33%	33.33%	36.67%	33.33%
	✓	6000	4	24000	80.00%	73.33%	70.00%	53.33%	40.00%	33.33%
Summary 5x		0	6	0	30.00%	26.67%	26.67%	26.67%	20.00%	6.67%
	✓	1000	6	6000	56.67%	53.33%	33.33%	40.00%	43.33%	33.33%
	✓	2000	6	12000	53.33%	60.00%	43.33%	53.33%	43.33%	33.33%
	✓	4000	6	24000	70.00%	53.33%	60.00%	53.33%	46.67%	46.67%

Table 3: Complete experiments table