# An Empirical Study of LLM Reasoning Ability Under Strict Output Length Constraint

Anonymous ACL submission

## Abstract

Recent work has demonstrated the remarkable potential of Large Language Models (LLMs) in test-time scaling. By making models think before answering, they are able to achieve much higher accuracy with extra inference computation. However, in many real-world scenarios, models are used under time constraints, where an answer should be given within a certain output length. It is unclear whether and how the reasoning ability of different LLMs remain effective under strict constraints. We take a first look at this problem by conducting an in-depth empirical study. Specifically, we test 30 LLMs on common reasoning datasets under a wide range of output length budgets, and we analyze the correlation between the inference accuracy and various properties including model type, model size, prompt style, etc. We also consider the mappings between token budgets and actual on-device latency budgets. The results have demonstrated several interesting findings regarding the budget-aware LLM reasoning ability that differ from the unconstrained situation, e.g. the optimal choices of either model size or prompt style change under different budgets. These findings offer timely evaluation to this area and practical guidance for users to deploy LLMs under real-world latency constraints.

## 1 Introduction

With the rapid advancement of Large Language Models (LLMs), there is a growing interest in their capabilities in tasks requiring advanced reasoning, such as programming, mathematical problem solving, and complex decision making. Their reasoning ability has become an important factor in the deployment of LLMs in real-world applications.

Various methods have been proposed to enhance the reasoning ability of LLMs. Some of them focus on equipping models with human-like cognitive processes and behaviors, such as Chain-of-Thought (CoT) reasoning (Wei et al., 2022), self-correction (Kamoi et al., 2024), and multi-agent debating (Liang et al., 2024; Du et al., 2024) mechanisms. Other approaches further enhance LLMs' reasoning by allocating more computational resources at test time to encourage deeper thinking, as seen in methods like self-consistency (Wang et al., 2023) and best-of-N decoding (Lightman et al., 2024). OpenAI o1 (OpenAI, 2024) and its open-source replicas, such as QwQ (Team, 2024c) and Sky-T1 (Team, 2025) exemplify the integration of these approaches, using strategies like problem decomposition, multi-perspective reasoning, and error tracing to improve reasoning performance.

These methods boost LLMs' reasoning performance, but they also lead to lengthy reasoning steps, which incur considerable computation costs. Recent works are beginning to explore strategies to optimize or control this cost, aiming to strike a balance between performance and reasoning efficiency (Han et al., 2024; Chen et al., 2024b; Damani et al., 2024; Wang et al., 2025). Approaches include dynamically adjusting the number of reasoning steps based on task difficulty (Manvi et al., 2024; Li et al., 2024), imposing token limits in prompts to encourage concise responses (Han et al., 2024), and conducting token-aware training to incorporate length constraints at the embedding level (Takase and Okazaki, 2019; Butcher et al., 2024).

However, prior research neglects scenarios in which LLMs' reasoning may be constrained by output limits. We believe this is an important setting that deserves more attention. First, many real-world AI applications are time constrained, requiring rapid or even real-time decisions. For example, autonomous driving systems should make precise action predictions within a limited time frame (Wang et al., 2024). Second, time-constrained reasoning under deadlines is an important trait of human intelligence. It is interesting to study whether and how LLMs preserve their reasoning ability under strict output length constraints.

Therefore, we conduct an empirical study of open-source LLMs' reasoning ability with **strict** output length constraints. Specifically, we test different models on various math datasets, while limiting the number of output tokens. This ensures that the inference process can be guaranteed to complete within a certain time budget, and the achieved accuracy can be regarded as the actual performance of models in time-constrained settings. We further analyze whether and how different factors (model type, model size and prompt style) can affect such performance.

To evaluate LLMs' reasoning under constrained scenarios, the naive approach is to directly terminate the generation process at the maximum token budget. However, this approach may lead to poor performance that does not reflect models' real capability, because LLMs may not explicitly output final answers during the reasoning. Instead, we adopt a more reasonable scheme named early stopping, where reasoning is interrupted at several tokens before budgets. A termination message *"Time's Up! Therefore, the final answer is:"* is appended to the end of model output, inducing LLMs to generate final answers in a structured format during the continued inference, until the generated tokens reaching the total budget. By avoiding abrupt reasoning truncation, this scheme faithfully reflects LLMs' time-constrained reasoning capabilities. We will elaborate on these two methods in Section 3.

Our experiments lead to several interesting and even surprising findings in Section 5. For example, when testing LLMs under output constraints, we observe the disagreement with scaling law (Kaplan et al., 2020) and the change in golden models and prompts in different deployment settings. Carefully tailored reasoning models are also not necessarily better than normal instruction tuned models. When mapping token budget to latency budget on real devices, we find that medium sized models often achieve the best efficiency under strict latency limits, while larger models gradually surpass them as latency constraints are relaxed. We expect these findings to present a general impression of how existing LLMs perform under strict output length constraints, and give practitioners some useful guidance to deploy LLMs in time-sensitive scenarios.

Our contributions are as follows:

1. We study an important scenario of LLM reasoning. To the best of our knowledge, this is the first thorough empirical study of LLM reasoning under strict output length constraints.
2. We conduct extensive experiments with a wide range of LLMs of various sizes and types. We evaluate their reasoning ability across mathematical datasets of varying difficulty.
3. We summarize several interesting findings, which may be helpful for researchers to understand the time-constrained reasoning ability of LLMs and improve them accordingly, and for developers to make informed choices based on their deployment scenarios.

Our code and data will be fully open-sourced.

## 2 Related Work

**Test-Time Scaling**. Rather than expanding model parameters and training data (Kaplan et al., 2020), recent studies now emphasize test-time scaling to improve LLMs' reasoning capabilities (OpenAI, 2024). This can be achieved by methods like repeated sampling (Snell et al., 2024; Brown et al., 2024), sequential sampling (Lee et al., 2025; Hou et al., 2025), and tree-based search (Hao et al., 2023; Chen et al., 2024a; Yao et al., 2023). Furthermore, researchers began to explore training LLMs using reinforcement learning to think deeper and generate longer CoTs (OpenAI, 2024; DeepSeek-AI et al., 2025). Despite of improvement of these methods, LLMs' reasoning ability under strict output length constraints is still unexplored.

**Efficient Reasoning Techniques.** Several works (Nayab et al., 2025; Han et al., 2024) showcase that adding output length limits into prompts can encourage LLMs to generate more concise yet still correct responses. Other works (Damani et al., 2024; Wang et al., 2025) allocate additional computation budget based on predicted complexity of queries, either by routing to larger models or conducting more samplings to vote for the final answer. Besides, other methods (Manvi et al., 2024; Li et al., 2024) allow for mid-generation control during multiple samplings to prune unpromising traces. The focus of our work is not to present an optimized method to improve LLMs' reasoning efficiency. Instead, our contribution is the rational and elaborate evaluation of LLMs reasoning ability under strict token budgets. Since there are evidence (Yuan et al., 2024; Xu et al., 2025) that directly adding length limits into prompts for LLMs to follow often fails or even brings performance degeneration, we do not adopt this method to impose strict output length constraints for LLMs' reasoing.

## 3 Method

We use two methods to impose output length constraints on LLMs' reasoning, as shown in Figure 1. They are referred to as **directly terminating** and **early stopping**. Both methods can strictly ensure the generated tokens do not exceed given budgets.

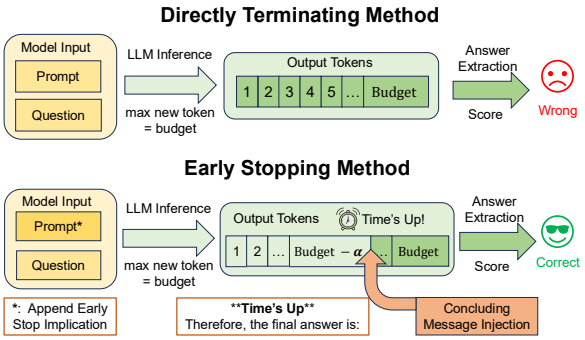


Figure 1: Two methods used in our work to ensure strict output length constraints for LLM reasoning.

### 3.1 Directly Terminating at Token Budget

The first method is to directly terminate LLMs' reasoning at token budgets. This can be done by setting the parameter *max_new_token* in LLM inference APIs, such as vLLM (Kwon et al., 2023), to the token budget. The final answer is then extracted from all generated tokens. Details of the extraction and scoring procedures are provided in Appendix A.1

This method is very straightforward but prone to underestimation of LLMs' reasoning ability under constrained scenarios, because abrupt truncation at token budgets during the reasoning process may result in incomplete responses. Only the problems whose responses fall within the token budgets can be correctly solved. As token budgets increase, more problems will be solvable within the budgets, and the accuracy across the dataset will also improve.

### 3.2 Early Stopping Before Token Budget

To obtain more reliable understanding of LLMs' reasoning capability under output constraints, we propose to early stop reasoning before token budgets and instruct LLMs to conclude final answers at once. As shown in Figure 1, this needs two modifications to directly terminating: appending early stop implication in model input construction and early stopping with concluding message injection.

Before inference, we append the model prompt with some words to inform LLMs of the potential early stopping during the reasoning process. The keyword `**Time's Up!**` is used as the signal of early stopping. During LLMs' reasoning, when the number of generated tokens reaches $Budget - \alpha$, we will append concluding message with signal `**Time's Up!**` in it, at the end of model outputs. Then LLMs are allowed to conclude their answers within $\alpha$[1] tokens, thus ensuring the total output length is still within token budgets. The final answer will be extracted and scored from tokens generated after concluding message injection. To ensure fairness, the procedures used here are the same as directly terminating method. If LLMs can finish their output generation using less than $Budget - \alpha$ tokens, then concluding message injection and further inference are not needed. And the final answer will be derived from all output tokens, just like the directly terminating method.

We list the full version of token budget implication and concluding message in Appendix A.2.

## 4 Experiment Setup

In this section, we present details of experiment settings. These include the evaluated datasets, models and prompt styles, and the evaluation framework.

### 4.1 Datasets

To evaluate LLM's reasoning ability, we use the test splits of two math datasets: GSM8K (Cobbe et al., 2021) and MATH500 (Lightman et al., 2023).

**GSM8K** includes 8.5k grade school level math word questions with high linguistic diversity, designed to test LLMs' ability to perform step-by-step reasoning. Its test split consists of 1319 problems.

**MATH500** is a scale extraction from the original MATH (Hendrycks et al., 2021b) dataset, with high school to early college level math problems. It includes subtopics like algebra, geometry and number theory, designed to evaluate advanced mathematical reasoning and problem-solving skills of LLMs. There are 500 problems in total for testing.

### 4.2 Models

To enrich the evaluation within our work, we selected three types of open-sourced LLMs:

**Instruction models** include series like Qwen-2.5-Instruct (Team, 2024b; Yang et al., 2024a), Phi-3-128k-instruct (Abdin et al., 2024a), gemma-2-it (Team, 2024a), and Llama-3.2 (Meta, 2024). Other

[1]We set $\alpha = 25$ in all our experiments, which is large enough to cover the correct final answers of tested problems.

| Model | Size | GSM8K | MATH500 |
|---|---|---|---|
| DRD-Qwen | 1.5B | 75.7(75.7) | 68.2(71.4) ↑ |
| DRD-Qwen | 7B | 87.9(87.9) | 77.0(81.2) ↑ |
| DRD-Qwen | 14B | 92.0(92.0) | 78.6(87.4) ↑ |
| DRD-Qwen | 32B | 94.5(94.5) | 79.8(86.0) ↑ |
| QwQ | 32B | 95.5(95.5) | 82.0(87.6) ↑ |
| Sky-T1 | 32B | 96.4(96.4) | 87.6(87.6) |
| Qwen2.5-Math | 1.5B | 85.0 | 74.0 |
| Qwen2.5-Math | 7B | 95.5 | 82.4 |
| Mathstral | 7B | 83.6 | 51.2 |
| Qwen-2.5 | 1.5B | 73.9 | 53.0 |
| Qwen-2.5 | 3B | 85.7 | 65.8 |
| Qwen-2.5 | 7B | 91.9 | 75.6 |
| Qwen-2.5 | 14B | 94.8 | 79.0 |
| Qwen-2.5 | 32B | 95.9 | 81.0 |
| Qwen-2.5 | 72B | 95.8 | 83.2 |
| gemma-2 | 2B | 64.8 | 23.8 |
| gemma-2 | 9B | 87.7 | 48.0 |
| gemma-2 | 27B | 90.8 | 56.2 |
| Llama-3.2 | 1B | 48.8 | 26.6 |
| Llama-3.2 | 3B | 76.2 | 48.6 |
| Llama-3.1 | 8B | 84.1 | 46.2 |
| Llama-3.1 | 70B | 94.9 | 62.6 |
| Ministral | 8B | 87.1 | 56.8 |
| Mistral-Nemo | 12B | 85.6 | 43.6 |
| Mistral-Small | 22B | 91.7 | 61.2 |
| Phi-3-mini | 3.8B | 86.7 | 39.0 |
| Phi-3-small | 7B | 88.9 | 50.8 |
| Phi-3-medium | 14B | 88.0 | 49.6 |
| Phi-3.5-mini | 3.8B | 86.8 | 45.2 |
| Phi-4 | 14B | 95.1 | 79.2 |

Table 1: Accuracy of tested models on both datasets. LLMs are prompted with step by step style. Max new token for non reasoning models: 4096, for reasoning models: 4096 (8192). The ↑ sign means reasoning models' score can still increase if more tokens are allowed.

models include Llama-3.1-8B-Instruct (Grattafiori et al., 2024) , Ministral-8B-Instruct-2410 (Mistral, 2024d), Mistral-Nemo-Instruct-2407 (Mistral, 2024b), Mistral-Small-Instruct-2409 (Mistral, 2024c) and Phi-4 (Abdin et al., 2024b).

**Math models** include those specially trained or fine tuned using mathematical data. We tested Qwen2.5-Math-Instruct (Yang et al., 2024b) and Mathstral-7B (Mistral, 2024a).

**Reasoning models** include QwQ-32B-Preview (Team, 2024c; Yang et al., 2024a), Sky-T1 (Team, 2025), DeepSeek-R1-Distill-Qwen series (DeepSeek-AI et al., 2025). They tend to generate longer reasoning steps than instruction or math models to scale up their problem solving ability.

### 4.3 Prompts

We use the following prompt styles to guide LLMs' reasoning in three different patterns:

1. **step-by-step (sbs).** This is the most common style to elicit model reasoning ability.
2. **coarse-to-fine (c2f).** This requests LLMs to give a coarse-grained reasoning summary before starting fine-grained reasoning steps.
3. **answer-and-verify (aav).** This style lets the models to give an initial answer quickly, then verify and revise it iteratively.

We expect c2f and aav styles can be used to better conclude final answers for LLMs if their reasoning is early stopped due to limited token budgets. We show the influence of prompt styles on LLMs' reasoning performance in Sec 5.2. The full version of prompt styles can be found in Appendix B.1. In order to align with the LLMs' training procedure, we construct model inputs based on their default chat templates and guidance from Hugging Face (Wolf et al., 2020). Model input construction process is also well illustrated in Appendix B.2.

### 4.4 Evaluation Framework

We use the evaluation framework from Qwen-2.5-Math (Yang et al., 2024b), which supports our datasets and models. Zero-shot and greedy decoding strategy is used to guarantee performance consistency. Package version used in experiments is transformers (Wolf et al., 2020) 4.46.3 and vLLM (Kwon et al., 2023) 0.6.3.post1. To give an outline of models' basic reasoning ability on the evaluation framework, and simplify the reference to their names in following sections, we report their performance on both datasets in Table 1.

## 5 Results and Findings

Through comprehensive evaluation and analysis of experiment results, we conclude five interesting findings, which lead to future directions worthy of exploration and some practical guidance for deploying LLMs under strict output length constraints. We also build a website[2] to show some examples, allowing readers to gain a clearer understanding when reading the following findings.

### 5.1 Early Stopping Method Outperforms Direct Terminating

**Finding 1:** Compared to directly terminating at token budget, early stopping consistently improves LLMs' reasoning performance on all combinations of datasets (GSM8K and MATH500) and prompt styles (sbs, c2f, and aav) in our evaluation.

[2]Examples can be found in https://time-is-up.github.io/

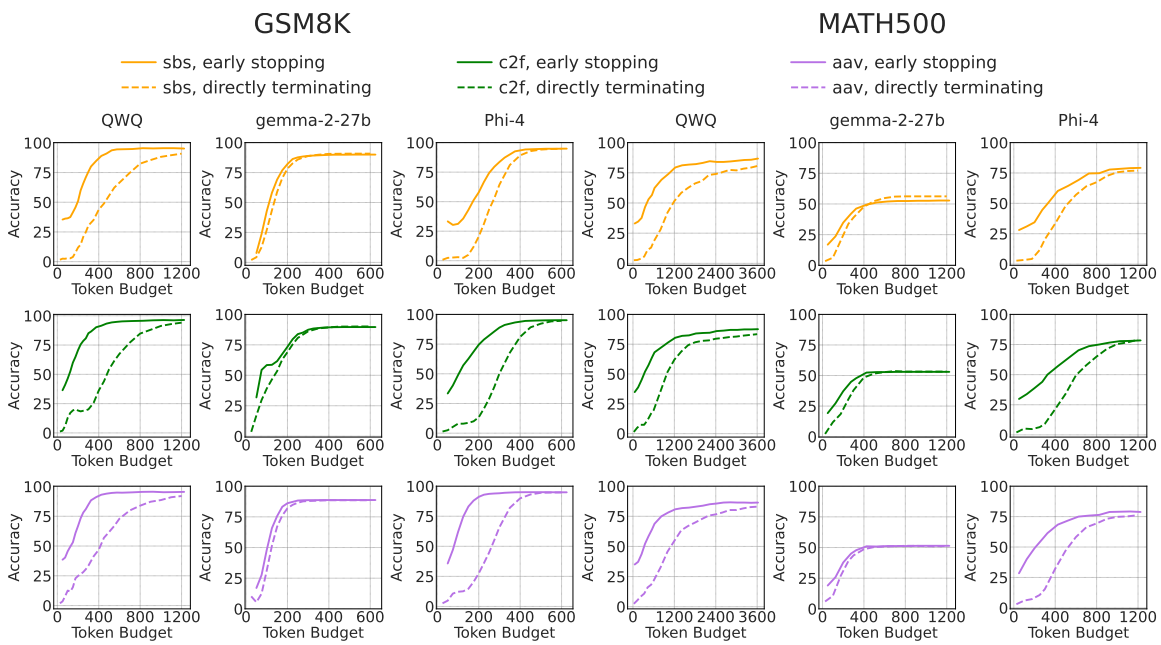


Figure 2: Early stopping method (solid line) outperforms directly terminating (dashed line) on GSM8K (left) and MATH500 (right) datasets. Prompting style: sbs, c2f and aav.

| Model | GSM8K | | | | | | MATH500 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Budget 50 | | | Budget 150 | | | Budget 100 | | | Budget 200 | | |
| | aav | c2f | sbs | aav | c2f | sbs | aav | c2f | sbs | aav | c2f | sbs |
| Sky-T1 | **46.6** | 44.6 | 38.4 | **83.7** | 70.0 | 58.5 | 40.0 | **40.4** | 36.6 | **53.6** | 49.8 | 43.0 |
| QwQ | 40.3 | **41.0** | 36.2 | 60.9 | **64.9** | 45.7 | 37.0 | **38.6** | 34.8 | 42.6 | **44.6** | 35.6 |
| DRD-Qwen-1.5B | 9.7 | **11.8** | 9.9 | **48.6** | 48.4 | 45.8 | **19.0** | 17.0 | 18.2 | 32.4 | **33.0** | 32.2 |
| DRD-Qwen-7B | 27.4 | **28.0** | 24.8 | **67.4** | 49.2 | 65.1 | **29.6** | 28.2 | 29.4 | 36.8 | 28.8 | **40.4** |
| DRD-Qwen-14B | 27.5 | **29.1** | 27.7 | **68.2** | 66.0 | 62.9 | **32.4** | 29.4 | 30.8 | **41.8** | 40.8 | 34.4 |
| DRD-Qwen-32B | 35.7 | **36.2** | 32.6 | **76.7** | 72.9 | 72.3 | **38.4** | 35.8 | 36.6 | **47.4** | 43.4 | 46.0 |
| Qwen2.5-Math-1.5B | **16.7** | 15.0 | 15.9 | **31.9** | 31.1 | 31.5 | **24.6** | 22.4 | 22.0 | **31.6** | 31.4 | 31.5 |
| Qwen2.5-Math-7B | 29.4 | **30.9** | 29.6 | **49.1** | 47.7 | 46.7 | **37.0** | **37.0** | 36.4 | **43.8** | **43.8** | 42.0 |
| Mathstral-7B | **19.7** | 16.8 | 8.6 | 54.3 | **63.5** | 47.8 | **22.6** | 21.8 | 16.8 | **35.4** | 33.0 | 31.0 |
| Qwen-2.5-1.5B | **21.2** | 19.8 | 9.8 | 43.4 | **49.6** | 25.7 | 18.2 | **19.0** | 15.4 | **27.0** | 26.8 | 25.2 |
| Qwen-2.5-3B | 13.4 | **21.5** | 12.4 | 35.0 | **49.7** | 36.0 | 22.0 | **23.6** | 21.0 | 31.8 | **32.0** | 29.6 |
| Qwen-2.5-7B | **46.4** | 35.0 | 20.5 | **78.2** | 60.1 | 48.1 | **39.2** | 31.6 | 26.6 | **50.0** | 43.4 | 35.4 |
| Qwen-2.5-14B | **46.3** | 40.4 | 24.3 | **82.8** | 65.4 | 39.0 | **44.0** | 39.2 | 25.2 | **55.8** | 50.4 | 38.0 |
| Qwen-2.5-32B | **57.6** | 47.8 | 36.2 | **88.8** | 76.1 | 56.3 | **50.2** | 44.0 | 38.4 | **60.4** | 53.2 | 46.2 |
| Qwen-2.5-72B | **52.0** | 43.4 | 37.0 | **85.6** | 75.6 | 57.8 | **48.4** | 45.0 | 44.0 | **58.0** | 54.0 | 48.8 |
| Ministral-8B | **25.8** | 21.0 | 6.7 | 58.8 | **69.3** | 44.7 | 23.8 | **25.8** | 15.6 | 35.4 | **39.2** | 28.4 |
| Mistral-Nemo | **24.2** | 20.1 | 5.7 | **65.0** | 58.5 | 44.7 | **22.6** | 21.8 | 15.2 | 31.4 | **31.6** | 26.0 |
| Mistral-Small | **31.8** | 23.1 | 6.6 | **72.6** | 70.1 | 45.5 | **31.2** | 26.2 | 16.8 | **41.2** | 38.6 | 29.8 |
| gemma-2-2b | 9.9 | **11.0** | 6.4 | 42.5 | 34.5 | **46.8** | **12.6** | 11.0 | 7.2 | **17.8** | 14.8 | 16.2 |
| gemma-2-9b | **39.8** | 37.5 | 14.3 | 67.9 | 61.1 | **70.8** | **23.2** | 22.4 | 17.2 | **35.6** | 33.0 | 34.8 |
| gemma-2-27b | 27.6 | **54.4** | 23.7 | **83.2** | 67.5 | 76.9 | 25.8 | **27.2** | 23.6 | 40.4 | **41.2** | 37.0 |
| Phi-3-mini | **22.5** | **22.5** | 20.1 | 51.6 | **71.9** | 59.1 | 21.0 | **21.6** | 22.0 | 30.8 | **36.8** | 31.2 |
| Phi-3.5-mini | **15.2** | 13.0 | 6.7 | **58.8** | 54.0 | 40.2 | **20.0** | 14.8 | 15.2 | **31.4** | 29.2 | 30.2 |
| Phi-3-small | **47.4** | 41.2 | 22.5 | **78.5** | 76.6 | 71.6 | **30.4** | 30.2 | 27.0 | **40.4** | 39.0 | 39.4 |
| Phi-3-medium | **31.5** | 27.1 | 10.3 | **70.0** | 58.6 | 39.5 | **28.0** | 26.4 | 25.2 | 36.8 | 35.8 | **37.4** |
| Phi-4 | **47.1** | 40.2 | 30.3 | **88.0** | 68.6 | 51.2 | **40.0** | 33.8 | 31.0 | **52.2** | 41.6 | 39.0 |
| Llama-3.2-1B | **3.4** | 3.2 | 3.3 | 28.0 | **28.7** | 28.1 | 8.8 | 8.4 | **9.4** | **17.4** | 14.6 | 15.8 |
| Llama-3.2-3B | **19.6** | 15.8 | 10.8 | 54.1 | 38.9 | **55.6** | **18.2** | 15.4 | 16.8 | **29.6** | 23.4 | 25.8 |
| Llama-3.1-8B | **35.2** | 21.8 | 12.1 | **69.8** | 51.3 | 58.9 | **22.4** | 17.4 | 16.6 | **30.8** | 25.2 | 26.0 |
| Llama-3.1-70B | **51.7** | 42.1 | 23.2 | **82.5** | 69.1 | 67.9 | **34.0** | 27.0 | 25.0 | **45.2** | 45.0 | 41.0 |

Table 2: In most cases, c2f and aav prompt styles outperform sbs under strict token budgets. The highest accuracy for each model at each budget is highlighted in bold.

We illustrate the performance of both methods from three models in Figure 2, and plot results of all tested models in Appendix C.1. From these figures, we find the advantage on performance of early stopping method can last till its accuracy converges or crosses with the accuracy of directly terminating method. We also notice the difference of two methods on models' convergent accuracy. On GSM8K dataset, it is negligible, however on MATH500 this difference can be up to 5%.

Terminating exactly at token budget truncates LLMs' reasoning process, leading to none or incorrect answer extraction. However, early stopping and concluding method can help LLMs to generate correct final answers even when partial reasoning steps are available Therefore, in order to fully illustrate and study LLMs' reasoning capabilities under token budgets, we only report the results of early stopping method in the rest of this paper.

### 5.2 Prompt and Thinking Pattern Matters

**Finding 2:** While the one-fits-all optimal prompt style doesn't exist, coarse-to-fine (c2f) and answer-and-verify (aav) outperform step-by-step (sbs) in most scenarios under output length constraint.

Table 2 demonstrates the superiority of aav and c2f styles at different token budgets. Figures of complete results can be found in Appendix C.2. Combining with the examination of LLMs' responses, we surmise that the better LLMs can understand and follow the implication in prompts and the more lengthy reasoning steps they tend to generate, the more likely c2f and aav styles can help increase LLMs' performance than sbs style.

For example, we find that Qwen2.5-Math models can not follow c2f and aav formatted style, thus generating reasoning steps very much like sbs style. DeepSeek-R1-Distill models generate responses in the correct style, but they all start with a thinking trace wrapped between <think> and </think>. Therefore, we observe negligible improvement on these models when switching prompts to encourage models to output brief analysis or speculative answers at early stage of their reasoning process.

Compared with other models on the same dataset, QwQ, Qwen-2.5 (7, 14, 32B) and Phi-4 models have more performance improvement when prompted in c2f or aav style. We assume this stems from the fact that they tend to generate more reasoning steps, thus more token budget is needed to achieve accuracy convergence when prompted in sbs style (over 400 tokens on GSM8K and 1k tokens on MATH500). Therefore, their responses may have more redundancy and can be compressed in a more concise way when prompted in c2f or aav style, which results in higher possibility of correct answer derivation under token budget.

More difficult dataset, MATH500, requires more lengthy and more *informative* steps to get the problems solved. So the benefits of answering questions prematurely by compressing reasoning steps will decrease, which leads to limited improvement of c2f and aav styles over sbs.

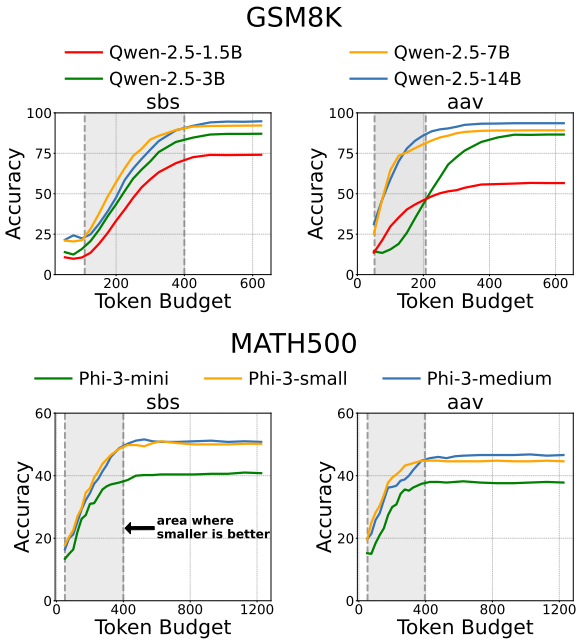


Figure 3: Under token budget, small models can outperform larger models on both datasets.

### 5.3 Larger Models are NOT Always Better

**Finding 3:** Under strict output constraints, the reasoning performance of LLMs might not scale monotonically with the model size. In other words, larger is not always better.

The most representative examples for this finding are the Qwen-2.5 and Phi-3 series. As shown in Figure 3, the highlighted parts in gray background is the area where anomalies exist. On the GSM8K dataset, within the token budget from 100 to 400 and prompted in sbs style, the accuracy of Qwen-2.5-7B model is consistently higher than 14B. Similar phenomenon also exists in aav

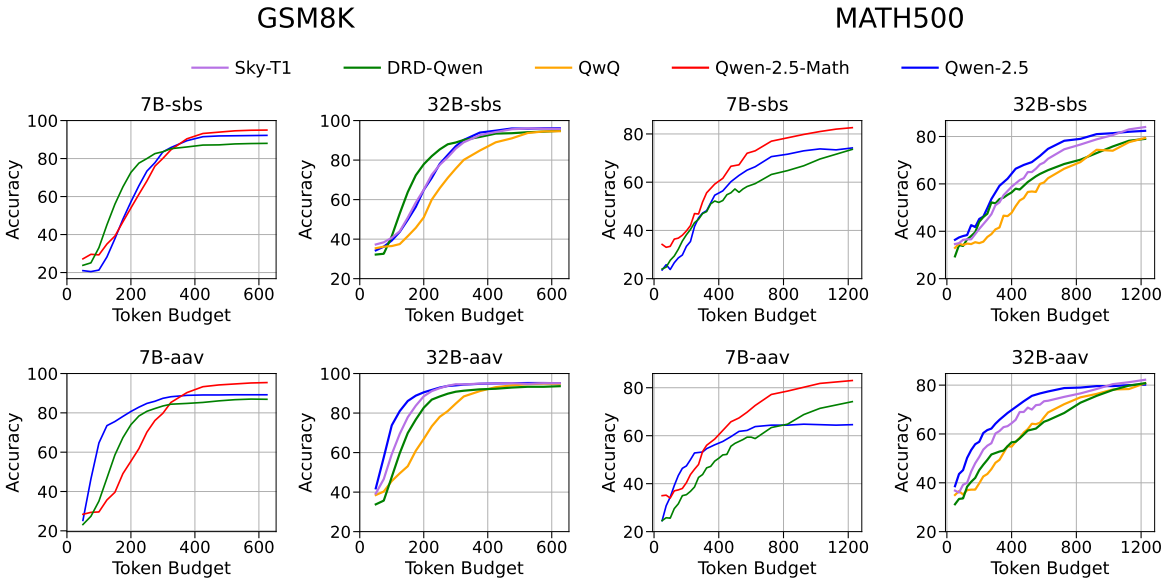


Figure 4: Under token budgets, reasoning models are not always better than instruction tuned or math models.

prompt style under token budget of 200 for Qwen-2.5-7B versus 14B, and Qwen-2,5-1.5B versus 3B. As for Phi-3 series on MATH500, 7B (Phi-3-small) model matches or even surpass 14B (Phi-3-medium) model under token budget of 400. Considering the 50% saving in size of smaller models from above pairwise comparison, their advantage over larger models is quite surprising. We plot all examples in Appendix C.3.

After checking the responses of Qwen models, we find that smaller Qwen models (7B/1.5B) require fewer output tokens to complete GSM8K problems than their larger counterparts (14B/3B). For instance, the median and average output token length for 7B model are 14.6% and 14.2% smaller than 14B, tested using sbs style and directly terminating method with budget set to 4096. On GSM8K, both small and large models can achieve high accuracy of completed questions, but they are prone to make mistakes when forced to early stop and conclude. Thus large models perform worse than small ones under strict token budgets.

One possible explanation for the abnormal phenomenon of Phi-3 model series can be found in its technical report (Abdin et al., 2024a), which states that Phi-3-medium model is trained on the same amount of data with Phi-3-small but for slightly more epochs. The improvement from 7B to 14B is not as significant as that from 3.8B to 7B on several benchmarks. Our experiments from Table 1, Figure 14 and Figure 15 also indicate that they have similar capability on both GSM8K and MATH500 datasets. This implies that trained on the same amount of data, LLM's reasoning ability under strict token budget may be diluted by over-large model size.

Finding 3 inspires us to reconsider the relationship between LLMs' reasoning capability and parameter size. Although larger models exhibit stronger capabilities on most benchmarks under no output limit, they may underperform smaller models on the ability to conduct precise and concise reasoning under strict constraint of token budgets.

### 5.4 Reasoning Models are NOT Always Better

**Finding 4:** Under strict output length constraints, reasoning models don't always outperform instruction tuned or math models.

In Figure 4, on GSM8K, DRD-Qwen models (7B, 32B) perform the best within token budgets smaller than 300, prompted in sbs style. But when using aav style, Qwen-2.5 instruction tuned models become the best choices, as they output speculated answers at early stage of reasoning, which helps to conclude final answers under strict token budgets. On MATH500, Qwen2.5-Math-7B model performs the best among models of 7B size under token budgets between 300 and 1.2k. And Qwen-2.5 32B instruction tuned model consistently performs the best among models of 32B size within token budget of 1000, prompted in both sbs and aav styles.

For easy questions, LLMs can solve them using a few simple reasoning steps, thus the conciseness and length of responses are more important. Therefore, instruction tuned models can beat rea-

soning models when prompted with aav style to derive correct answers more efficiently. For complex questions, reasoning models tend to generate much longer reasoning steps than other models, which can be identified from Figure 2 and Appendix C.1. Although reasoning models can achieve higher accuracies when token budget is large enough, their accuracy curves rise more slowly at early stages and can not match those of instruction tuned or math models under strict output token budgets.

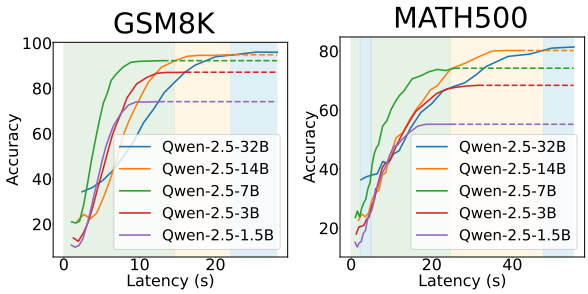


Figure 5: Comparison of Qwen-2.5-Instruct models on NVIDIA A800 GPU under infernce latency budget.

### 5.5 Mid-Sized Models are Latency-Optimal

**Finding 5:** In latency critical scenarios that require on-device reasoning, we should prioritize middle-sized models, even if there is enough resource to deploy a larger model.

Given the specific device to deploy, the inference latency of LLMs is mainly composed of prefill time and decode time. The prefill latency is determined by model input length, and decode latency is the sum of latency of every new token, which is determined by the context length before generating each token. We first evaluate the input length for all scenarios in Table 3 from Appendix C.4, which shows that the input ranges within 150-250 tokens. Then we measure the correlation between latency and input and output length for all models and prompt styles on a single NVIDIA A800 GPU. From examples in Figure 19 we find that the impact of input length on total latency can be ignored. And for each model, there exists a clear linear mapping between output tokens and inference latency.

To find the optimal model size under latency budgets, we plot the performance of Qwen-2.5-Instruct models prompted in sbs style in Figure 5, using the latency mapping derived through on-device profiling. The background of each region is highlighted using the curve color of the model that dominates that area on accuracy. On both datasets, 7B model exhibits significant advantage within limited latency budgets. As the budget relaxes, the performance of 14B and 32B models surpasses 7B model. Results of other model series can be found in Figure 20 and Figure 21, where the general trend is similar. This finding implies when deploying LLMs on devices under strict latency constraints, we should prioritize middle sized models, with a typical value around 7B.

## 6 Discussion

Apart from math reasoning tasks, we also cautiously speculate that our findings can generalize to other similar reasoning domains. So we conduct extra experiments of Qwen-2.5 and DRD-Qwen model series on mmlu_stem and ACPBench (Kokel et al., 2025) datasets. mmlu_stem is a subset of STEM subjects (such as astronomy and biology) defined in MMLU (Hendrycks et al., 2021a). ACPBench contains both single and multi step reasoning tasks for evaluating actions and plans.

From experiment results, we have observed the performance improvement when switching sbs prompt style into aav or c2f for Qwen-2.5 models on mmlu_stem dataset, which adheres to our finding 2. However, this improvement is rather limited for Qwen-2.5 models on ACPBench. As for different model sizes, we find that Qwen-2.5 14B can outperform 32B on ACPBench when prompted with aav style. Qwen-2.5 1.5B also outperforms 3B on ACPBench when prompted with c2f. These results support our finding 3 that large models are not always better than smaller ones. We also compare the impact of model types of the same size. On both datasets, we notice notable advantage of instruction tuned models under token budgets within 1000, although reasoning models have higher accuracy when budget is relaxed to 4096.

## 7 Conclusion

We investigated the reasoning capabilities of large language models (LLMs) under time-constrained scenarios by imposing output token length limitations. Our findings reveal that the performance of LLMs can vary significantly depending on factors such as model size, architecture, and prompt design when operating under different constraints. We expect this work to shed light on this under-explored area and offer valuable insights for practitioners aiming to deploy LLMs in real-world and time-sensitive applications.

## Limitations

While our study provides a first step toward understanding LLM reasoning under time-constrained conditions, it has several limitations. First, we focused primarily on mathematical reasoning tasks, which, while representative, may not fully capture the diverse range of real-world applications where time constraints are critical. Future work should extend this analysis to other domains, such as programming and decision making. Second, our experiments were conducted using a limited set of LLMs and prompt designs, which may not comprehensively represent the broader landscape of available models and techniques. Third, the validity of our findings (e.g. the optimal model sizes under different time budgets) may be threatened by the different training procedures of each model, which are usually not transparent to researchers. Finally, our evaluation assumes a direct correlation between token budget and latency, which, though practical, does not account for hardware-specific variations in real-world deployment.

## References


Marah Abdin, Jyoti Aneja, Hany Awadalla, et al. 2024a. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

Marah Abdin, Jyoti Aneja, Harkirat Behl, et al. 2024b. Phi-4 technical report. *Preprint*, arXiv:2412.08905.

Bradley Brown, Jordan Juravsky, Ehrlich, and othe. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

Bradley Butcher, Michael O'Keefe, and James Titchener. 2024. Precise length control in large language models. *arXiv preprint arXiv:2412.11937*.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024a. Alphamath almost zero: process supervision without process. *arXiv preprint arXiv:2405.03553*.

Xingyu Chen, Jiahao Xu, Liang, et al. 2024b. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, et al. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Mehul Damani, Idan Shenfeld, Andi Peng, Andreea Bobu, and Jacob Andreas. 2024. Learning how hard to think: Input-adaptive allocation of lm computation. *Preprint*, arXiv:2410.04707.

DeepSeek-AI, Daya Guo, Dejian Yang, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2024. Token-budget-aware llm reasoning. *Preprint*, arXiv:2412.18547.

Shibo Hao, Yi Gu, Ma, et al. 2023. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, et al. 2021b. Measuring mathematical problem solving with the math dataset. *Preprint*, arXiv:2103.03874.

Zhenyu Hou, Xin Lv, Rui Lu, Zhang, et al. 2025. Advancing language model reasoning through reinforcement learning and inference scaling. *arXiv preprint arXiv:2501.11651*.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. 2024. When can LLMs actually correct their own mistakes? a critical survey of self-correction of LLMs. *Transactions of the Association for Computational Linguistics*, 12:1417–1440.

Jared Kaplan, Sam McCandlish, Henighan, et al. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Harsha Kokel, Michael Katz, Kavitha Srinivas, and Shirin Sohrabi. 2025. Acpbench: Reasoning about action, change, and planning. In *AAAI*. AAAI Press.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, et al. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Kuang-Huei Lee, Ian Fischer, Wu, et al. 2025. Evolving deeper llm thinking. *arXiv preprint arXiv:2501.09891*.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, et al. 2024. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In *The Twelfth International Conference on Learning Representations*.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. 2024. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17889–17904, Miami, Florida, USA. Association for Computational Linguistics.

Hunter Lightman, Vineet Kosaraju, Yura Burda, et al. 2023. Let's verify step by step. *Preprint*, arXiv:2305.20050.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, et al. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Rohin Manvi, Anikait Singh, and Stefano Ermon. 2024. Adaptive inference-time compute: Llms can predict if they can do better, even mid-generation. *Preprint*, arXiv:2410.02725.

Meta. 2024. Llama 3.2 model card.

Mistral. 2024a. Mathstral. https://mistral.ai/en/news/mathstral.

Mistral. 2024b. Mistral nemo. https://mistral.ai/en/news/mistral-nemo.

Mistral. 2024c. Mistral small. https://huggingface.co/mistralai/Mistral-Small-Instruct-2409.

Mistral. 2024d. Un ministral, des ministraux. https://mistral.ai/en/news/ministraux.

Sania Nayab, Giulio Rossolini, Marco Simoni, et al. 2025. Concise thoughts: Impact of output length on llm reasoning and cost. *Preprint*, arXiv:2407.19825.

OpenAI. 2024. Learning to reason with llms. OpenAI Blog.

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.

Sho Takase and Naoaki Okazaki. 2019. Positional encoding to control output sequence length. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3999–4004, Minneapolis, Minnesota. Association for Computational Linguistics.

Gemma Team. 2024a. Gemma.

NovaSky Team. 2025. Sky-t1: Fully open-source reasoning model with o1-preview performance in $450 budget. https://novasky-ai.github.io/posts/sky-t1. Accessed: 2025-01-09.

Qwen Team. 2024b. Qwen2.5: A party of foundation models.

Qwen Team. 2024c. Qwq: Reflect deeply on the boundaries of the unknown.

Tianqi Wang, Enze Xie, Ruihang Chu, Zhenguo Li, and Ping Luo. 2024. Drivecot: Integrating chain-of-thought reasoning with end-to-end driving. *Preprint*, arXiv:2403.16996.

Xinglin Wang, Shaoxiong Feng, Yiwei Li, et al. 2025. Make every penny count: Difficulty-adaptive self-consistency for cost-efficient reasoning. *Preprint*, arXiv:2408.13457.

Xuezhi Wang, Jason Wei, Dale Schuurmans, et al. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Schuurmans, et al. 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.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. 2025. Chain of draft: Thinking faster by writing less. *Preprint*, arXiv:2502.18600.

An Yang, Baosong Yang, Binyuan Hui, et al. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

An Yang, Beichen Zhang, Binyuan Hui, et al. 2024b. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Shunyu Yao, Dian Yu, Zhao, et al. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Weizhe Yuan, Ilia Kulikov, Ping Yu, Kyunghyun Cho, Sainbayar Sukhbaatar, Jason Weston, and Jing Xu. 2024. Following length constraints in instructions. *Preprint*, arXiv:2406.17744.

# A Method Details

## A.1 Answer Extraction and Scoring

Answer extraction relies on several pattern matching operations. This process searches for specific

formats, and in most cases LLMs will put their final answers within \\boxed{}. Other phrases like "the answer is", "final answer is" are also used to locate the answer after them. If no specific patterns are found, the default method is to extract the last numeric value from the string, which guarantees that answers can be extracted even when the format does not strictly adhere to predefined patterns.

After the initial extraction, the function undergoes several post-processing steps to clean and normalize the extracted answer. These steps include removing leading colons, trailing periods, and slashes. Moreover, the extracted answer is cleaned of unnecessary whitespace and, depending on the dataset, may have units removed.

Finally, results from some models are manually reviewed to correct any formatting errors or inconsistencies that may have been overlooked during the automated extraction and post-processing steps. This manual review process enhances the accuracy and reliability of the extracted answers, particularly in cases where the output string does not strictly follow expected patterns or where the automated extraction process might have introduced errors.

After the answer is extracted and cleaned, we use scoring method from framework Qwen-2.5-Math (Yang et al., 2024b). The accuracy is calculated based on both numerical and symbolic equality between the extracted answer and labeled answer from datasets.

### A.2 Token Budget Implication and Concluding Message

Here is the full version of token budget implication and concluding message used in early stopping method. Phrase `**Time's Up!**` is used as the signal to execute early stop, as shown in Figure 6.

**Token Budget Implication**

Notice: When you are interrupted by the keyword **Time's Up!**, stop reasoning immediately.
Based on your reasoning so far, conclude with:
Therefore, the final answer is: \\boxed{{[answer]}}.
Where [answer] is just the final number or expression that solves the problem.

**Concluding Message**

**Time's Up!**
Therefore, the final answer is:

Figure 6: Token budget implication and concluding message used in early stopping method.

## B Experiment Setup Details

### B.1 Prompt Style

We list the full version of step-by-step (sbs), coarse-to-fine (c2f), and answer-and-verify (aav) prompt styles in Figure 7. Step-by-step style requires the LLM to follow a direct linear reasoning process. Coarse-to-fine starts with a brief initial answer (coarse) and then adds detailed reasoning (fine). Answer-and-verify begins with an intuitive answer, followed by verification and correction through detailed reasoning.

**Step by step (sbs) prompt style**

Please reason step by step.
Conclude with:
Therefore, the final answer is: \\boxed{{[answer]}}.
Where [answer] is just the final number or expression that solves the problem.

**Coarse to fine (c2f) prompt style**

Use the following pattern to solve the problem:
**Coarse-Grained Reasoning**
Provide a brief analysis and initial answer, focusing on efficiency and conciseness.

**Fine-Grained Reasoning**
Provide detailed reasoning step by step and a refined answer, focusing on correctness and rigor.

Conclude with:
Therefore, the final answer is: \\boxed{{[answer]}}.
Where [answer] is just the final number or expression that solves the problem.

**Answer and verify (aav) prompt style**

Use the following pattern to solve the problem:
**Quick Answer**
Provide an initial answer based on intuition or quick calculation.

**Verification**
Provide a revised answer through reasoning step by step. Correct previous mistakes, if any.

Conclude with:
Therefore, the final answer is: \\boxed{{[answer]}}.
Where [answer] is just the final number or expression that solves the problem.

Figure 7: Three different prompt styles used in experiments.

**Chat Templates**

```
"mistral_format": "<s>[INST] {system_message}\n\n{input}[/INST]",
"qwen_format": "<|im_start|>system\n{system_message}<|im_end|>\n<|im_start|>user\n{input}<|im_end|>\n<|im_start|>
assistant\n",
"phi3mini_format": "<|system|>\n{system_message}<|end|>\n<|user|>\n{input}<|end|>\n<|assistant|>\n",
"phi3small_format": "<|endoftext|><|system|>\n{system_message}<|end|>\n<|user|>\n{input}<|end|>\n<|assistant|>\n",
"phi3medium_format": "<|user|>\n{input}<|end|>\n<|assistant|>\n",
"phi4_format": "<|im_start|>system<|im_sep|>{system_message}<|im_end|><|im_start|>user<|im_sep|>{input}<|im_end|>
<|im_start|>assistant<|im_sep|>",
"llama_format": "<|begin_of_text|><|start_header_id|>system<|end_header_id|>\n\n{system_message}<|eot_id|>
<|start_header_id|>user<|end_header_id|>\n\n{input}<|eot_id|><|start_header_id|>assistant<|end_header_id|>\n\n",
"gemma_format": "<bos><start_of_turn>user\n{input}<end_of_turn>\n<start_of_turn>model\n",
"deepseek-r1-distill_format" : "<|begin_of_sentence|><|User|>{input}<|Assistant|>"
```

Figure 8: Chat templates of different model series tested in our experiments.

### B.2 Model Input Construction

Figure 8 shows the chat templates we use for all models in our experiments. The {system_message} will be replaced by prompt styles like sbs, c2f or aav, and {input} will be replaced by math problems. If the template does support system role, we concatenate prompt styles and problem with "\n\n" and then feed them into {input}. Figure 9 shows an example of constructing model input based on Qwen-2.5 chat templates and sbs prompt style.

### B.3 Discussion About License or Terms for Scientific Artifacts

All datasets, models and evaluation framework in this work are strictly used for academic and research purposes only and do not involve any commercial applications. Their usage is fully compliant with their respective licenses and the intended use specified by the original providers. No modifications or derivatives of them have been used in ways that would conflict with the terms set forth by the original licenses. The data and models have been used solely within the scope of this research and will not be deployed in any commercial or non-research contexts.

## C Evaluation Details

### C.1 Finding 1

Here we present a comparative analysis of various methods utilizing direct terminating (dashed lines) and early stopping (solid lines) on GSM8K and MATH500 datasets. The methods are evaluated across different prompting styles: step-by-step (sbs, Figure 10,11), coarse-to-fine (c2f, Figure 12,13), and answer-and-verify (aav, Figure 14,15). Each prompting style illustrates the performance differs from different models under varying prompting strategies, showing which strategies are more effective on specific datasets.

### C.2 Finding 2

Here we present a comparative analysis of various models' performance utilizing different prompting styles on the GSM8K (Figure 16) and MATH500 (Figure 17) datasets. The models are evaluated across step-by-step solution (sbs), coarse-to-fine (c2f), and answer-and-verify (aav). Each prompting style illustrates how the performance differs between different models under these 3 prompting strategies.

### C.3 Finding 3

Figure 18 presents an analysis of the Qwen-2.5-Instruct and Phi-3 models' performance on the GSM8K and MATH500 datasets under different token budgets. This analysis explores how the reasoning performance of large language models (LLMs) may not scale monotonically with the model size under certain output constraints. In other words, larger models do not always perform better. The figures illustrate this phenomenon by showing the performance variations of the models with different token budgets on both datasets.

### C.4 Finding 5

Here we list the median of input token length for all models in various dataset and prompt style sce-

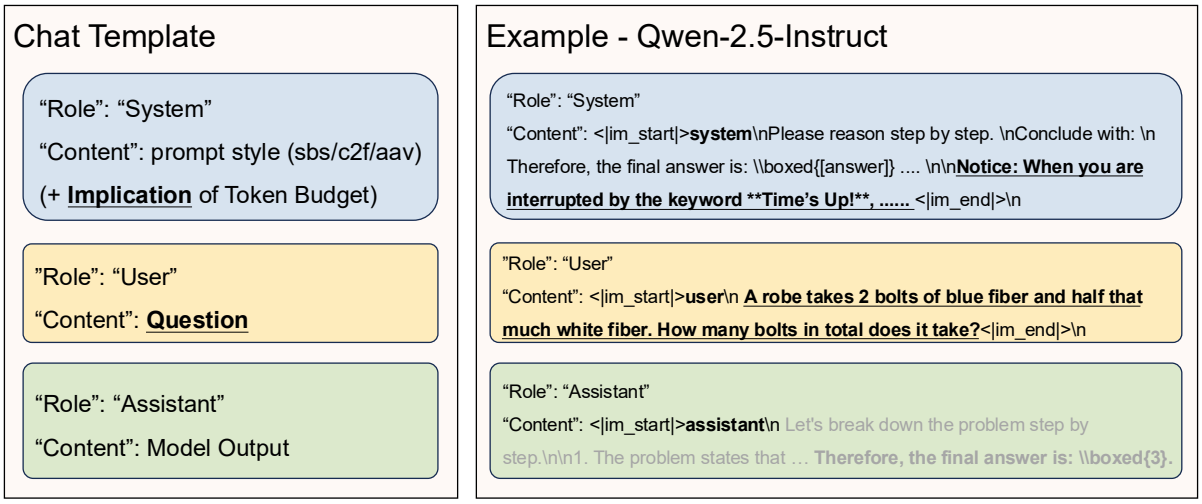


Figure 9: The chat template and an example of constructing model input.

| Model Series | GSM8K | | | MATH500 | | |
|---|---|---|---|---|---|---|
| | sbs | c2f | aav | sbs | c2f | aav |
| Mistral | 185 | 251 | 234 | 175 | 241 | 224 |
| Qwen-2.5 | 170 | 222 | 207 | 163 | 215 | 200 |
| Phi-3-mini | 184 | 254 | 235 | 172 | 242 | 223 |
| Phi-3-small | 167 | 219 | 204 | 161 | 213 | 198 |
| Phi-3-medium | 184 | 254 | 235 | 172 | 242 | 223 |
| Phi-4 | 166 | 218 | 203 | 160 | 212 | 197 |
| Llama-3.2 | 171 | 223 | 208 | 165 | 217 | 202 |
| gemma-2 | 176 | 229 | 217 | 169 | 222 | 210 |
| DRD-Qwen | 163 | 215 | 200 | 156 | 208 | 193 |

Table 3: The median of input token counts for different model types across various datasets and prompt styles.

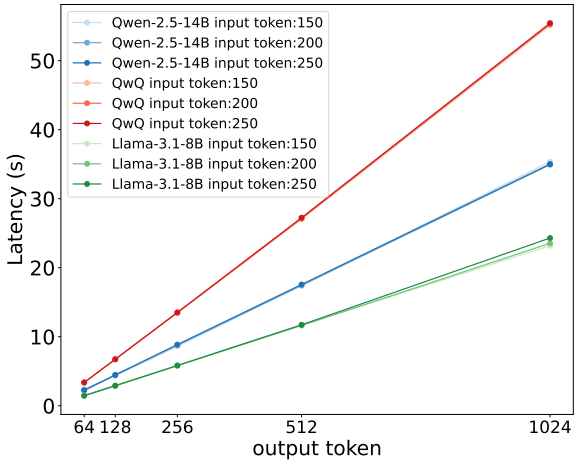


Figure 19: The mapping between output tokens and inference latency. The impact of different input length is quite negligible. And the inference latency is almost linearly correlated to the number of output tokens.

narios. Model inputs are formatted following the construction in Section B.2. For each model series, we use one of the models' tokenizer to encode all inputs. The results are shown in Table 3.

In most cases, the number of input token length falls within the range of 150-250. Therefore, we tested the mapping between output token and inference latency for three models under input token counts: 150, 200, and 250, as shown in Figure 19.

The results indicate that, within output token of 1024, the impact of input length on latency mapping is negligible. Consequently, we applied mapping calculation with input token as 200 in Section 5.5. In Figure 20 and 21, we show the evaluation of the Qwen-2.5, Phi-3, gemma-2, Llama-3.2, and DRD-Qwen series on the GSM8K and MATH500 datasets, prompted in sbs, c2f, and aav styles.

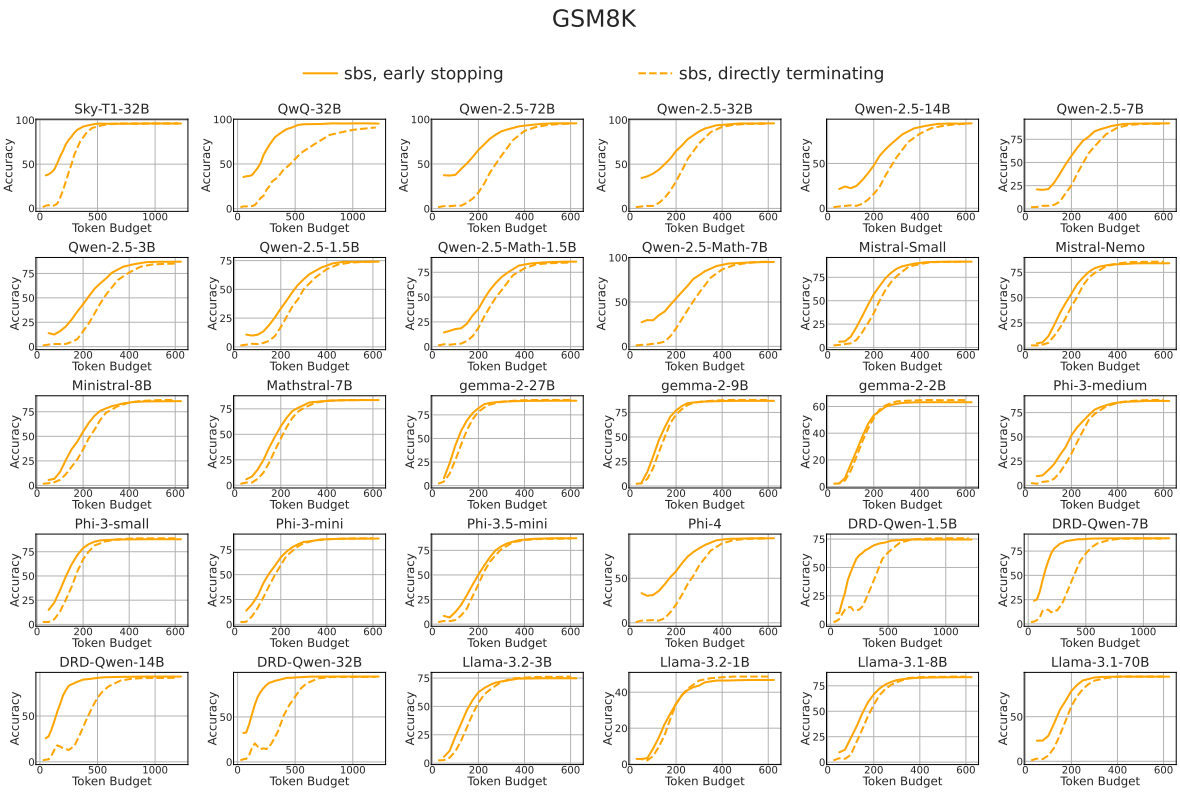


Figure 10: Early-stopping (solid line) outperforms directly terminating (dashed line) method on GSM8K datasets. Prompting style: sbs.

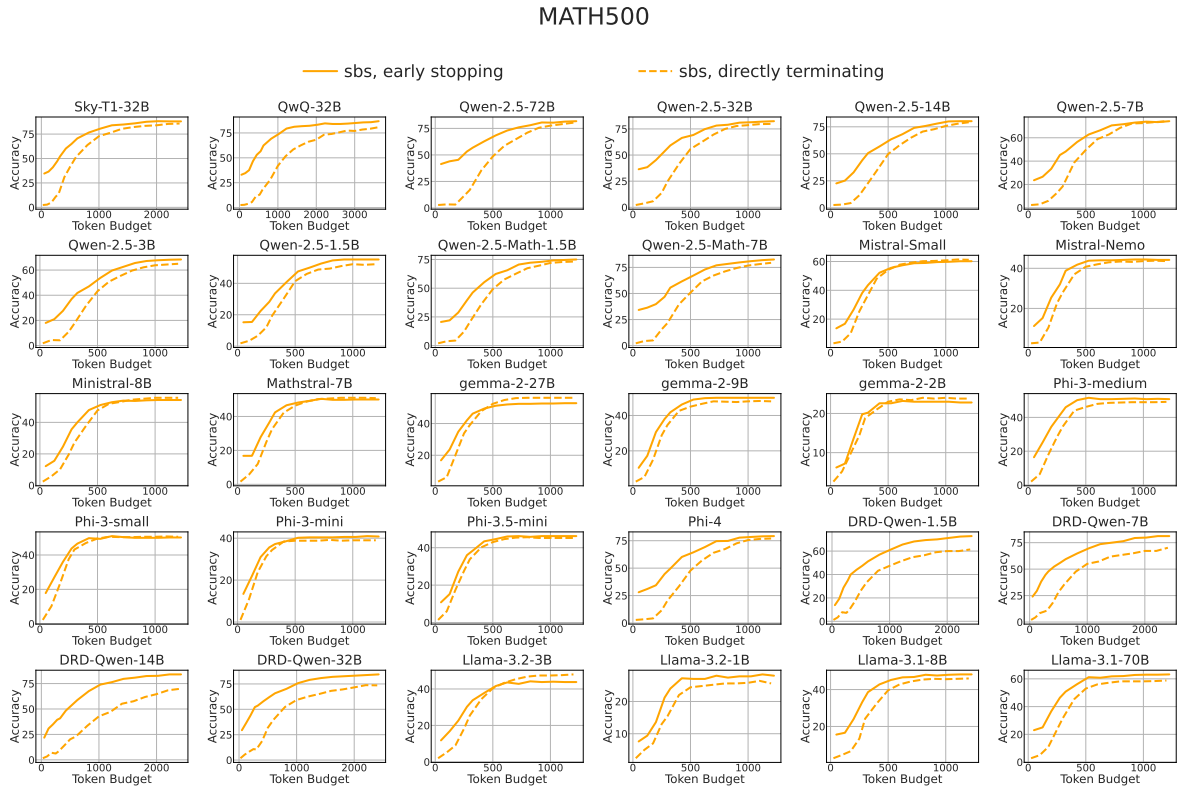


Figure 11: Early-stopping (solid line) outperforms directly terminating (dashed line) method on MATH500 datasets. Prompting style: sbs.

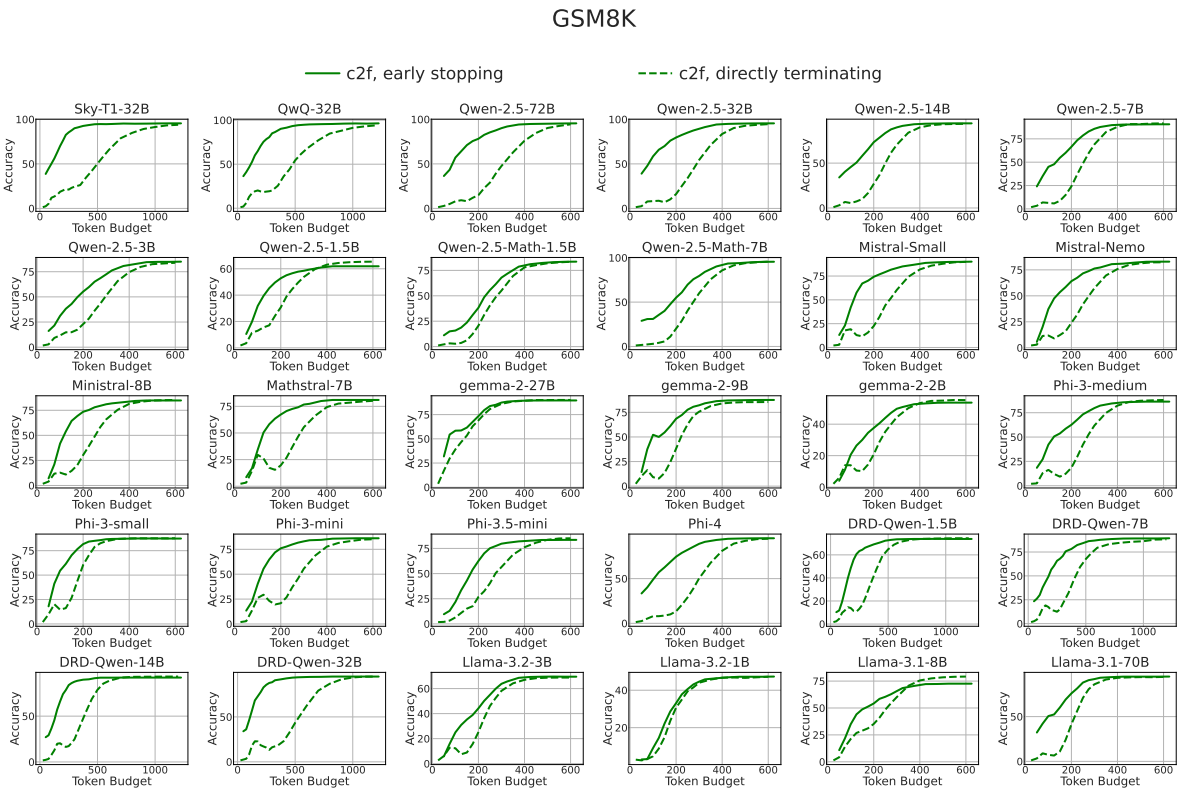


Figure 12: Early-stopping (solid line) outperforms directly terminating (dashed line) method on GSM8K datasets. Prompting style: c2f.

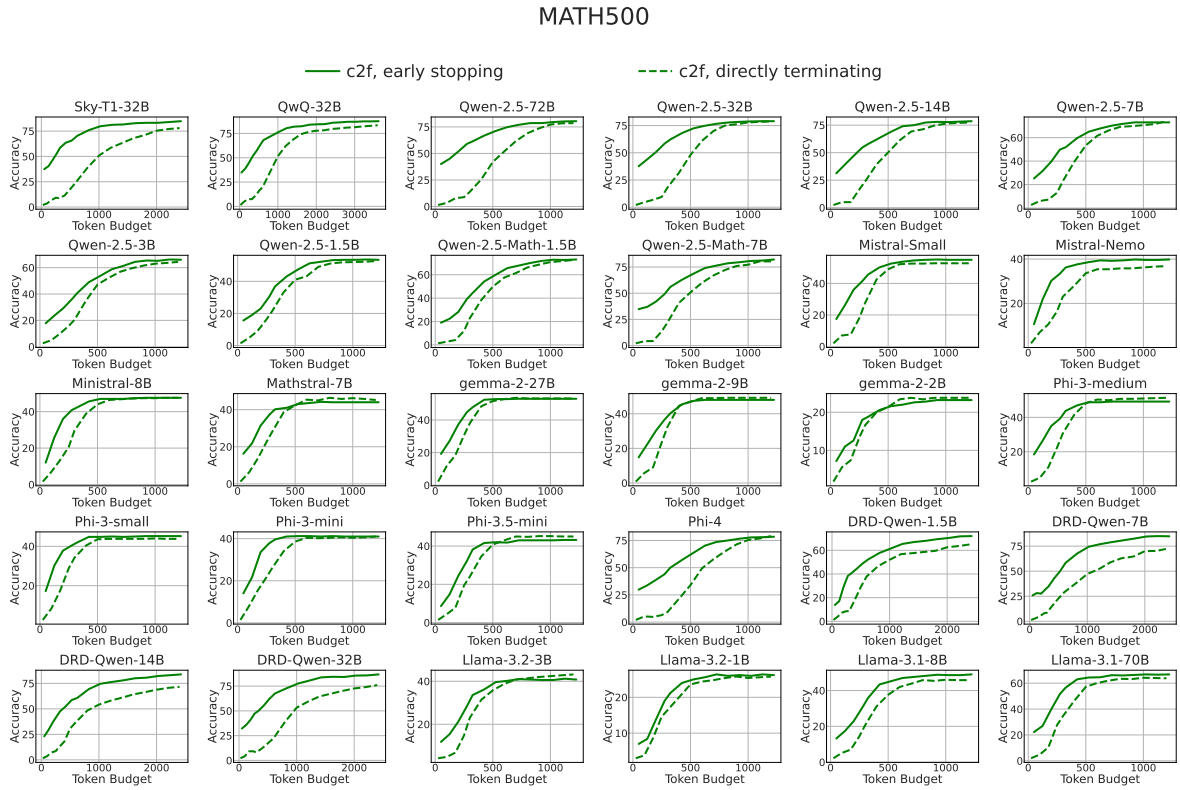


Figure 13: Early-stopping (solid line) outperforms directly terminating (dashed line) method on MATH500 datasets. Prompting style: c2f.

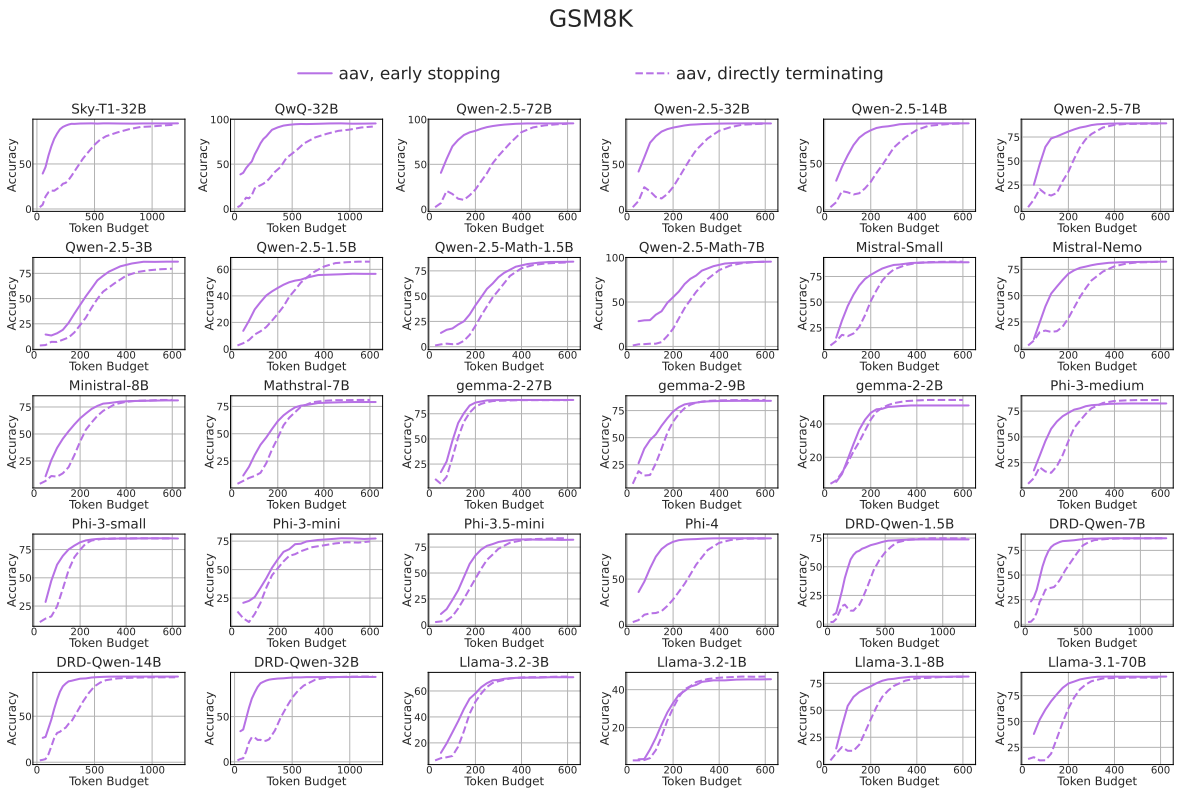


Figure 14: Early-stopping (solid line) outperforms directly terminating (dashed line) method on GSM8K datasets. Prompting style: aav.

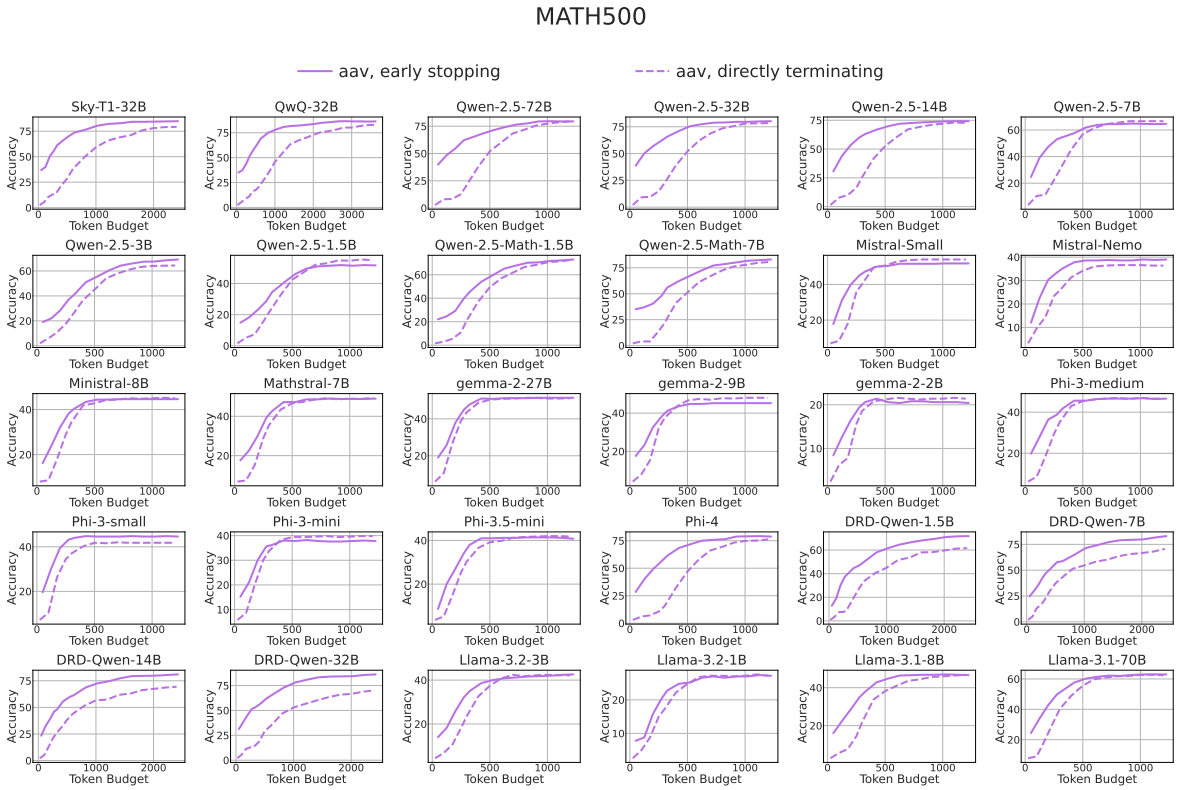


Figure 15: Early-stopping (solid line) outperforms directly terminating (dashed line) method on MATH500 datasets. Prompting style: aav.

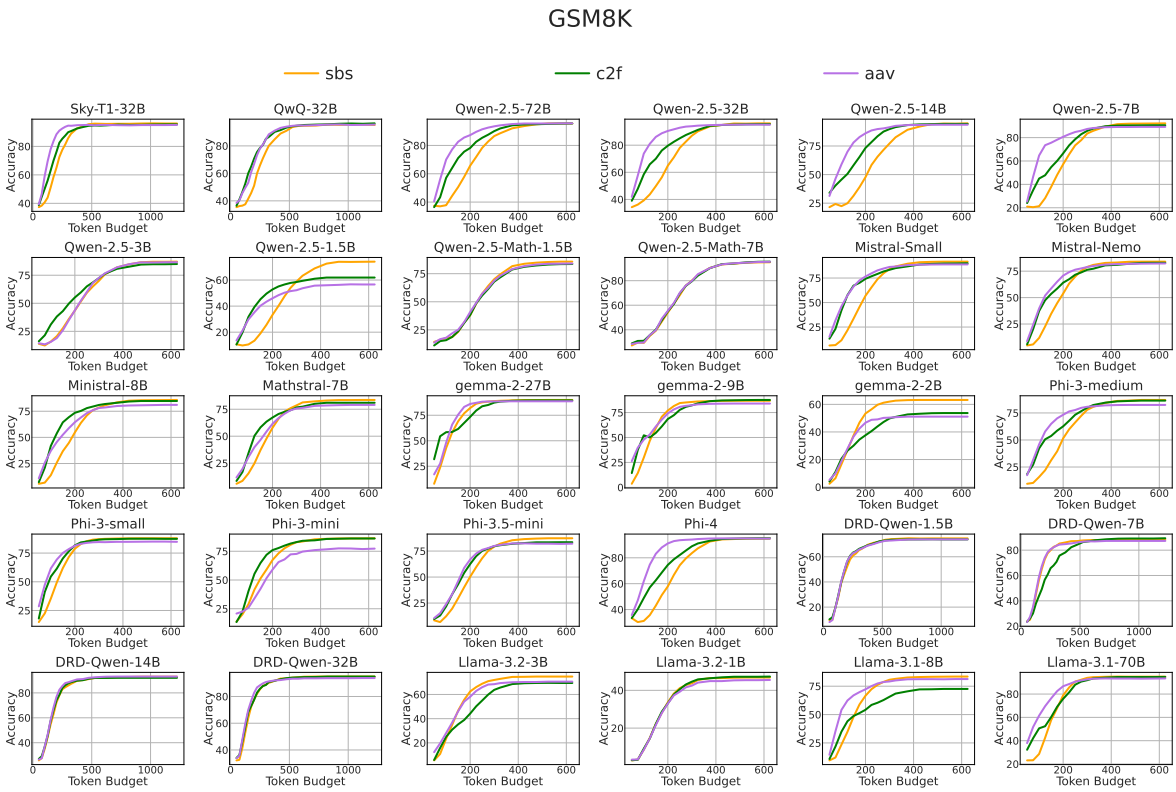


Figure 16: Comparison of different models' performance with early-stopping methods on GSM8K datasets. Prompting style: sbs, c2f and aav.

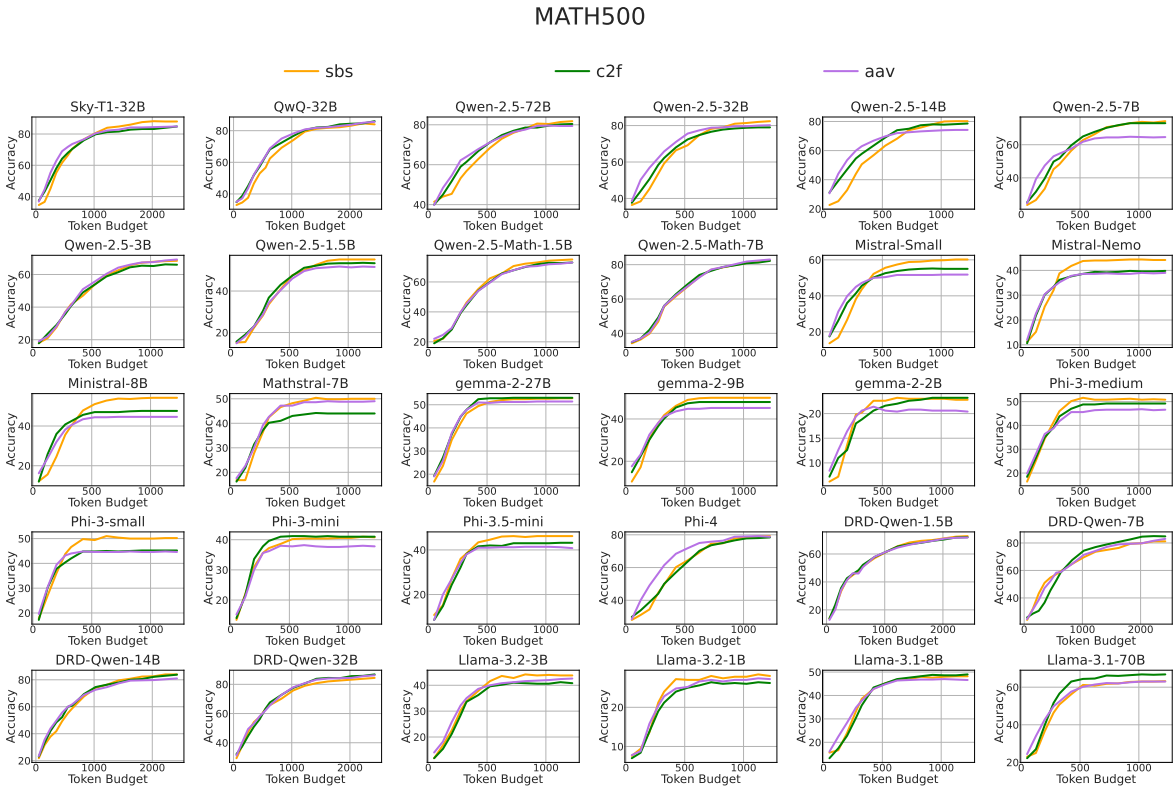


Figure 17: Comparison of different models' performance with early-stopping methods on MATH500 datasets. Prompting style: sbs, c2f and aav.

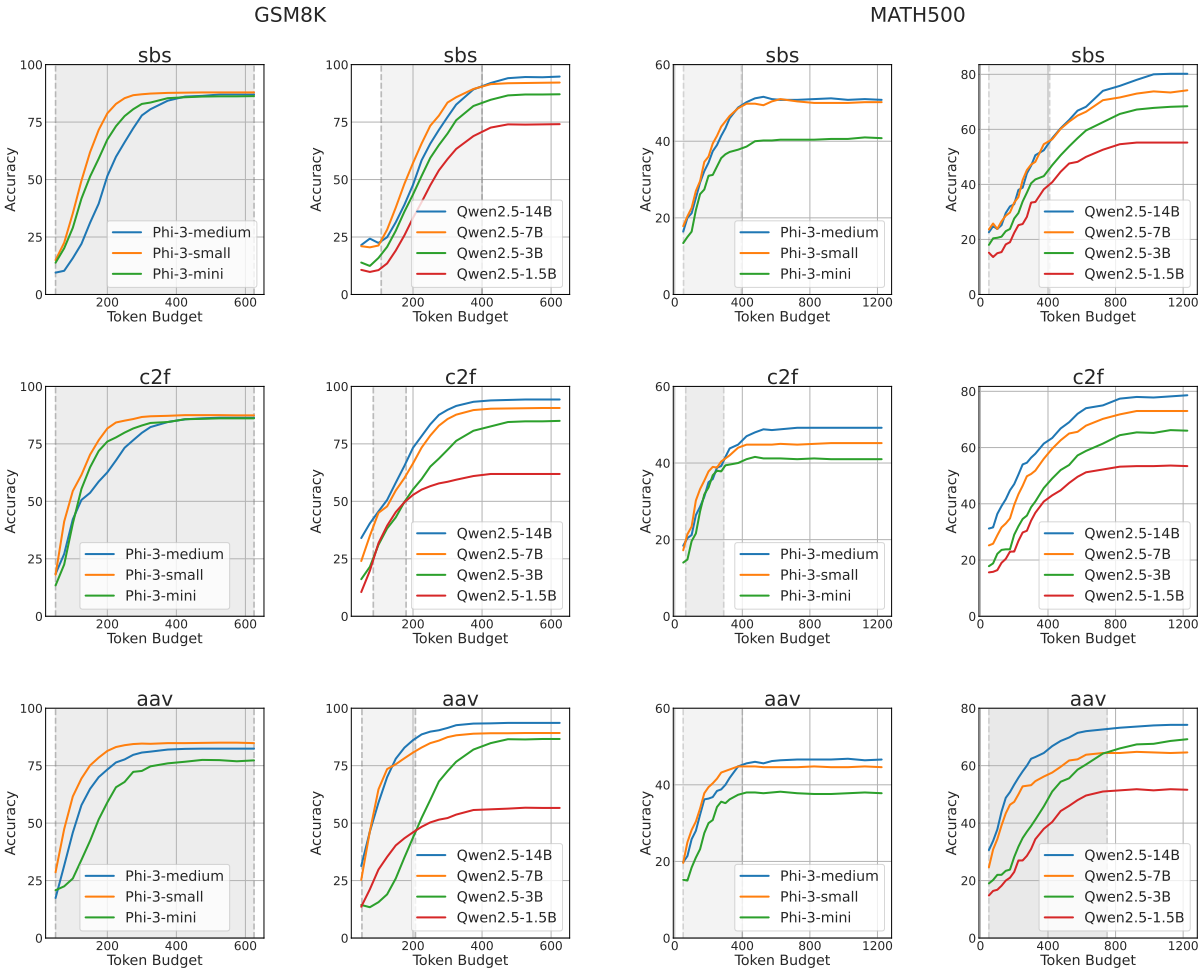


Figure 18: Qwen-2.5-Instruct and Phi-3 models' performance on GSM8K and MATH500 datasets with different token budgets.

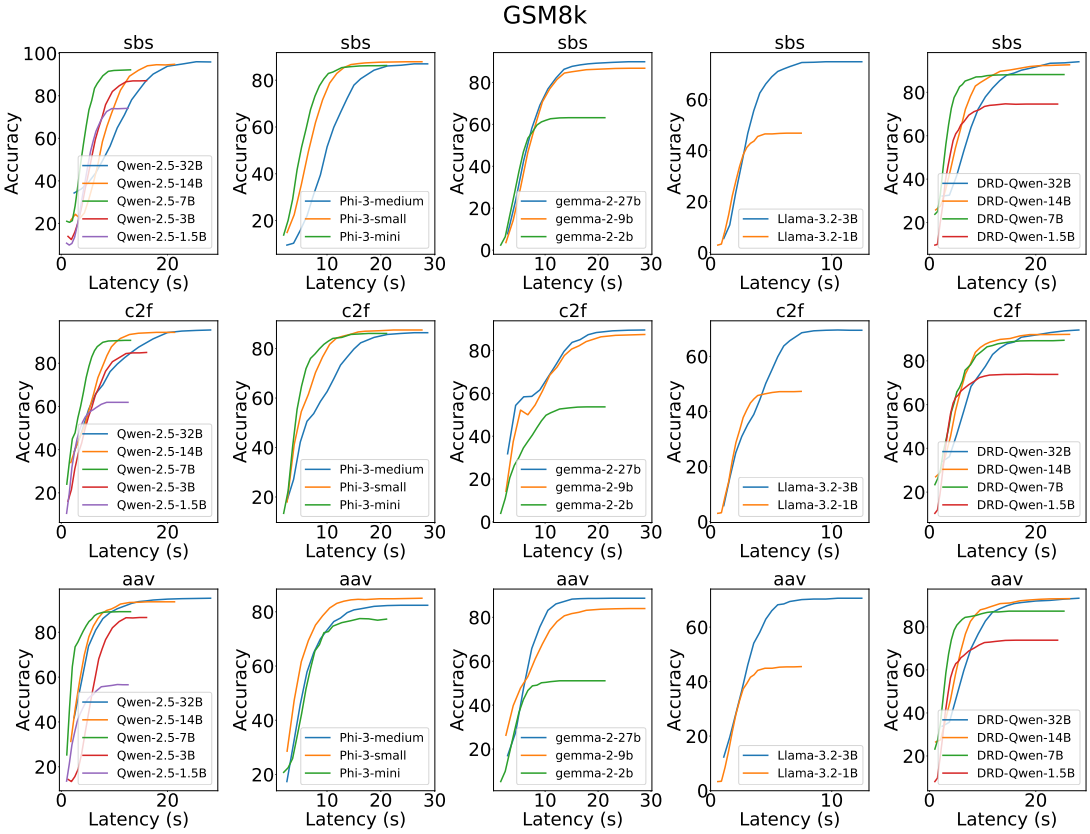


Figure 20: Models' performance under inference latency budget on GSM8K dataset.

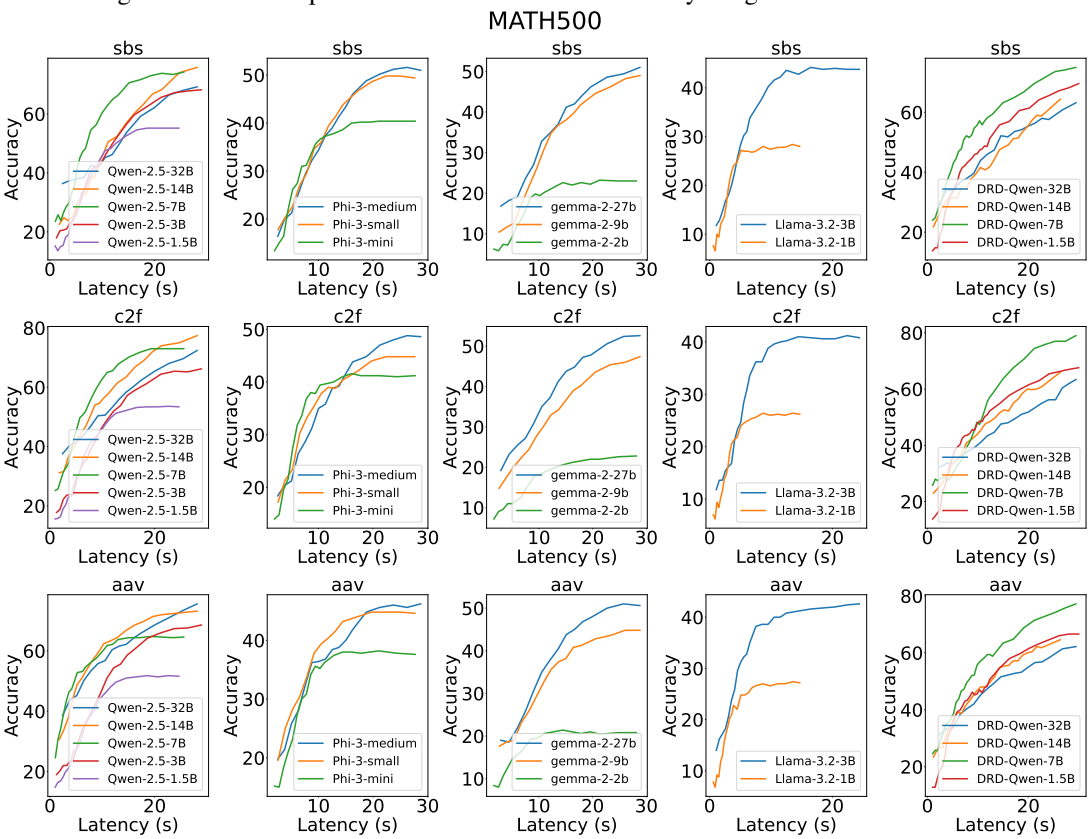


Figure 21: Models' performance under inference latency budget on MATH500 dataset.