Through the Valley: Path to Effective Long CoT Training for Small Language Models

Anonymous ACL submission

Abstract

Long chain-of-thought (CoT) supervision has become a common strategy to enhance reasoning in language models. While effective for large models, we identify a phenomenon we call *Long CoT Degradation*, in which small language models (SLMs; $\leq$3B parameters) trained on limited long CoT data experience significant performance deterioration. Through extensive experiments on the Qwen2.5, LLaMA3 and Gemma3 families, we demonstrate that this degradation is widespread across SLMs. In some settings, models trained on only 8k long CoT examples lose up to 75% of their original performance before fine-tuning. Strikingly, we further observe that for some particularly small models, even training on 220k long CoT examples fails to recover or surpass their original performance prior to fine-tuning. Our analysis attributes this effect to error accumulation: while longer responses increase the capacity for multi-step reasoning, they also amplify the risk of compounding mistakes. Furthermore, we find that *Long CoT Degradation* may negatively impacts downstream reinforcement learning (RL), although this can be alleviated by sufficiently scaled supervised fine-tuning (SFT). Our findings challenge common assumptions about the benefits of long CoT training for SLMs and offer practical guidance for building more effective small-scale reasoning models.

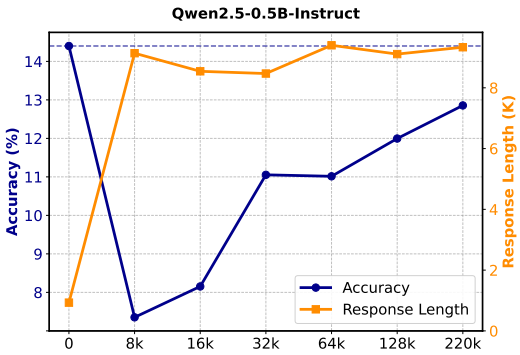

Figure 1: Accuracy and response length for Qwen2.5-0.5B across varying amounts of long CoT SFT data. Performance drops markedly at smaller data scales (8k-16k), even as response length increases significantly, indicating a critical failure mode in which the model generates longer but less accurate reasoning traces. We term this phenomenon *Long CoT Degradation*.

1 Introduction

Large reasoning models, such as OpenAI-o3 (OpenAI, 2025), Kimi-k1.5 (Team et al., 2025), and DeepSeek-R1 (Guo et al., 2025) have recently demonstrated impressive capabilities in complex reasoning tasks. A key strength of these models is their ability to generate long chain-of-thought (CoT) responses, which usually demonstrate advanced, reflective reasoning behaviors. These detailed reasoning responses, referred to as long CoT, constitute valuable resources for enhancing the reasoning ability of large language models (LLMs).

Despite the growing use of long CoT data in LLM training, there remains no consistent strategy for its integration. Current practices vary widely across models and objectives. For instance, some studies show that even supervised fine-tuning (SFT) with a relatively small amount of long CoT data (e.g., $\leq$10k examples) can effectively enhance reasoning capabilities in LLMs (Muennighoff et al., 2025; Xu et al., 2025; Ye et al., 2025). Others combine either limited or extensive long CoT SFT with subsequent RL training (Face, 2025; Chen et al., 2025; Bercovich et al., 2025; Wen et al.; Guo et al., 2025). However, in almost all these cases, the choice of data scale tends to be heuristic, and currently, there is limited empirical understanding of how the scale of long CoT data influences model performance across different training paradigms and different model sizes.

The call for a closer examination of this underexplored topic is especially pertinent given the growing interest in developing and deploying small

language models (SLMs), where strong reasoning capabilities are crucial due to their rising popularity and prevalence (Li et al., 2023; Hui et al., 2024; Lee et al., 2024; Agarwal et al., 2024). Compared to larger models, SLMs typically have limited capacity, which may affect their ability to generalize from verbose supervision, such as long CoT data (Feng et al., 2024). Although effective for large models, long CoT's verbosity may overwhelm smaller models, making it unclear to what extent SLMs can benefit from such training. Recent work even suggests that short CoT may be more effective for SLMs (Li et al., 2025a). However, these findings are based on relatively small-scale datasets (e.g., 8k examples), leaving it uncertain whether they hold at larger data scales.

In this paper, we conduct a systematic investigation into how the scale of long CoT data affects the performance of SLMs. Our results show that SLMs trained with small amounts of long CoT supervision (e.g., 8k to 16k examples) frequently suffer from substantial performance degradation (e.g., Fig. 1), a phenomenon we refer to as *Long CoT Degradation*. Building on this observation, we explore three central research questions: 1) How prevalent is *Long CoT Degradation*, and can SLMs recover from it? 2) What underlying mechanisms drive this degradation of SLMs? 3) Does long CoT degradation carry over to subsequent reinforcement learning (RL) stages, and can integrating long CoT SFT with RL mitigate its effects and improve overall training efficacy for SLMs?

To this end, we conduct extensive experiments across diverse model scales and families, ranging from 0.5B to 14B, confirming the prevalence of Long CoT Degradation. We next formulate hypotheses regarding its underlying causes and validate them through comprehensive experiments. Finally, we investigate the impact of long CoT SFT on the RL stage by analyzing the performance of three representative SLMs after RL.

Our contributions are threefold:

- **Empirical discovery of Long CoT Degradation:** To the best of our knowledge, we are the first to identify and systematically characterize the phenomenon of *Long CoT Degradation*, which arises when SLMs are trained with limited long CoT supervision. Our findings demonstrate that this degradation consistently occurs across a variety of model families and a wide range of SLM sizes, revealing a critical limitation in existing training practices. (§2)
- **Analysis of underlying mechanisms:** We attribute long CoT Degradation to error accumulation driven by length inflation in reasoning outputs. Our experiments reveal how insufficient long CoT supervision leads to disproportionately verbose and error-prone responses, ultimately harming performance. (§3)
- **Towards better training pipelines:** We examine how SFT with long CoT affects subsequent RL for SLMs. Our results show that while limited long CoT exposure may hinder RL performance, sufficiently scaled supervision during SFT can significantly boost both the efficiency and final performance of RL, even after the model is degraded. (§4)

2 Long CoT Degradation

This section presents a comprehensive empirical study on how models of varying sizes and families respond to long CoT SFT. We uncover a consistent degradation phenomenon caused by long CoT supervision and analyze the conditions under which models recover. In addition, we examine how continued exposure to long CoT affects the token efficiency of model outputs across sizes.

2.1 Terminology

In this work, we define **long CoT** as a reasoning process involving substantially longer sequences[1] that explicitly incorporate steps such as reflection, verification, and subproblem decomposition. Such chains are typically produced by large-scale reasoning models.

2.2 Experimental Setup

Datasets. We utilize the OpenR1-Math-220k dataset[2], comprising approximately 225,000 English math problems from NuminaMath 1.5 (LI et al., 2024), each paired with two to four verified reasoning traces generated by DeepSeek-R1. For training, we sample one correct trace per problem and exclude outliers exceeding 16,384 tokens (<1% of data). To analyze performance scaling, we construct six dataset subsets of increasing size: 8k, 16k, 32k, 64k, 128k, and 220k samples.

[1] For instance, the average length of long CoT responses in DeepSeek-R1-Zero approaches 10k tokens, whereas its base model averages fewer than 1k tokens (Guo et al., 2025).

[2] https://huggingface.co/datasets/open-r1/OpenR1-Math-220k, licensed under Apache 2.0.

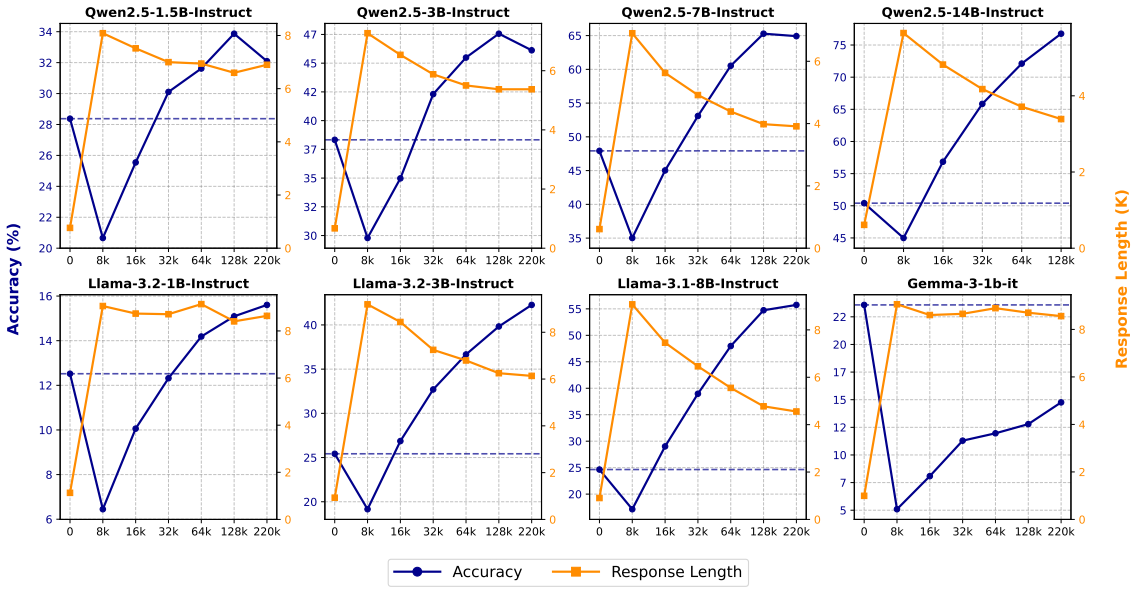

Figure 2: Comprehensive evaluation of multiple models trained with varying amounts of long CoT data. Accuracy is averaged across AIME24, AMC23, and MATH500, while response length is measured as the mean token count from 4,000 responses to MATH500. Per-benchmark results are provided in Appendix C.1.

Models. Our study considers nine instruct-tuned models from the Qwen, LLaMA, and Gemma model families, including Qwen-2.5 (0.5B, 1.5B, 3B, 7B, 14B) (Yang et al., 2024), LLaMA (3.2-1B, 3.2-3B, 3.1-8B) (Grattafiori et al., 2024), and Gemma-3-1B-it (Team, 2025).

Training Setup. Each model is fine-tuned using full-parameter SFT on each subset with consistent hyperparameters detailed in Appendix B. We use the LLaMA-Factory framework (Zheng et al., 2024) for training.

Evaluation Setup. We evaluate on three prevalent mathematical reasoning benchmarks: AIME24[3], AMC23[4], and MATH500 (Hendrycks et al., 2021). Generation length is capped at 16,384 tokens. Following the evaluation protocol of Guo et al. (2025), we use sampling with temperature 0.6 and top-p 0.95 to generate k responses per question ($k = 8$ for AIME24 and AMC23, $k = 4$ for MATH500)[5]. For each benchmark, we report the average accuracy computed over k responses (avg@k).

[3] https://huggingface.co/datasets/AI-MO/aimo-validation-aime

[4] https://huggingface.co/datasets/AI-MO/aimo-validation-amc

[5] Greedy decoding is avoided due to its high repetition rate and instability across checkpoints when evaluating long-output reasoning models (Guo et al., 2025).

2.3 Results

Degradation under long CoT supervision is prevalent, even in moderately sized models. Across all model families and sizes, we observe a notable drop in accuracy following exposure to long CoT SFT (Figures 1 and 2). Notably, the accuracy of Gemma3-1B-it falls to approximately 25% of its baseline after training on just 8k long CoT examples. Even the largest model in our study, Qwen2.5-14B-Instruct, suffers a drop from 50% to 45% accuracy. Moreover, this degradation is consistently accompanied by a sharp increase in response length. This suggests a deeper underlying issue in how models handle long CoT supervision, which we analyze further in Section 3.

All models exhibit recovery on more long CoT data, while larger models recover faster and more fully. Given the consistent performance degradation observed after training on 8k long CoT examples, we investigate whether models can recover as the number of training samples increases, and how recovery dynamics vary with model size. Figure 2 shows that larger models, such as Qwen2.5-7B and 14B, recover more quickly, requiring fewer additional examples before eventually surpassing their baseline performance. For instance, Qwen2.5-14B recovers and significantly exceeds its baseline after training on just 16k examples, while Qwen2.5-1.5B slightly surpasses its baseline with 32k examples. In contrast, as can

be seen in Figure 1 and 2, smaller models struggle to fully recover. Despite full exposure to 220k training examples, Qwen2.5-0.5B and Gemma-3-1B fail to reach their original baselines, with final accuracies dropping from 14% to 11% and from 24% to 15%, respectively. These results highlight that recovery from long CoT degradation is both data- and capacity-dependent.

Token efficiency improves with increased long CoT data and larger models. Figure 2 also reveals another interesting observation that increasing the number of long CoT training examples leads to improvements in both accuracy and reduced response length across all models, indicating a general gain in token efficiency. We suspect this is because models initially mimic superficial patterns in long CoT traces, resulting in verbose outputs. With more training, they gradually shift towards capturing the underlying reasoning structure, leading to shorter and more accurate responses. This effect is more evident in larger models. For instance, with training on 32k instances of long CoT data, Qwen2.5-14B-Instruct achieves an accuracy of 66% with an average response length of only 4k tokens, whereas its smaller counterpart, the 7B model, reaches only 53% accuracy despite producing longer responses averaging 5k tokens. This contrast suggests that larger models are more capable of leveraging long CoT to generate concise yet accurate answers. These findings motivate further investigation into the underlying mechanisms driving degradation and recovery, which we explore in the following section.

3 The Mechanism Behind Degradation

To better understand the phenomenon of Long CoT Degradation, we propose two hypotheses and design targeted experiments to empirically validate them.

3.1 Hypotheses

Our hypotheses are grounded in two lines of prior research. First, recent studies have examined system-2 reasoning in large language models, revealing phenomena such as multi-step reasoning and reflection (Xiang et al., 2025; Yu et al., 2024; Li et al., 2025b). While these behaviors are prevalent in long CoT data, their precise impact on the length and structure of reasoning chains remains unclear. Second, prior work on CoT has shown that as reasoning chains grow longer, the accumulation of intermediate mistakes increasingly undermines final output accuracy (Wu et al., 2025). However, analyses of error accumulation have thus far been largely limited to short CoT sequences and relatively small-scale models (e.g., variants of GPT-2 with fewer than 10 layers) (Wu et al., 2025), leaving open the question of whether similar mechanisms persist in longer, more real-world reasoning settings with modern LLMs.

Building on these insights, we aim to explain the degradation and recovery behaviors observed in our empirical study. To this end, we propose the following two hypotheses:

- **Hypothesis 1: Early adoption of surface-level reasoning patterns contributes to verbose outputs.** When exposed to a limited amount of long CoT supervision, SLMs rapidly adopt surface features of system-2 reasoning, such as reflection and multi-step structure. This early emergence of stylistic patterns is highly correlated with increased response length and may contribute to initial performance degradation, even before deeper reasoning skills are fully acquired.
- **Hypothesis 2: Longer outputs exacerbate error accumulation, reducing answer accuracy.** As output length increases, the reasoning process involves a greater number of steps, each of which introduces the potential for errors to propagate. Consequently, longer responses tend to accumulate more noise and irrelevant content, ultimately resulting in a noticeable decline in overall accuracy.

These two hypotheses are closely linked: the first aims to explain why models tend to generate long responses under long CoT supervision, while the second aims to account for the resulting drop in accuracy. Using these hypotheses as guidance, we further conduct empirical analyses in the following subsections.

3.2 Reflection Behavior Analysis

To validate Hypothesis 1, we examine whether models rapidly acquire surface-level features of system-2 reasoning, particularly reflective behavior, during early stages of fine-tuning on long CoT data. Our goal is to determine whether the emergence of such patterns coincides with the increase in response length observed in Section 2.3.

Setup. We identify reflective behavior in model outputs using a cross-validation approach follow-

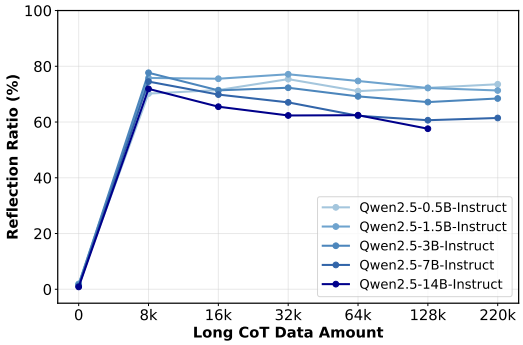

Figure 3: Reflection ratios of Qwen models of different sizes trained on varying amounts of long CoT data. The reflection ratio refers to the proportion of model responses (out of 4,000 on the MATH500 benchmark) that exhibit reflective behavior, as identified through cross-validation.

ing Liu et al. (2025). To robustly detect self-reflection in generated responses, we use two independent methods: 1) a keyword-based approach that labels a response as reflective if it contains any curated indicative keywords or phrases, and 2) an LLM-based approach, where GPT-4o-mini (Hurst et al., 2024) is prompted to determine whether the response exhibits reflective behavior. This dual-criteria strategy helps reduce false positives and enhances reliability. Full implementation details and prompt templates are provided in Appendix B.4.

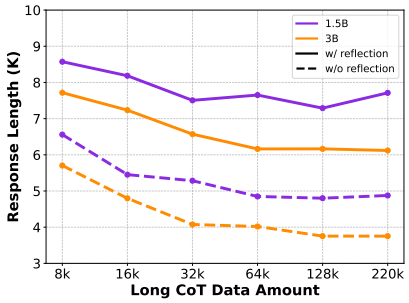

Figure 4: Average response lengths of Qwen2.5-{1.5B, 3B}-Instruct models trained with varying amounts of long CoT data. Solid lines represent responses exhibiting reflection behavior; dashed lines denote responses without reflection. Results for more models are provided in Appendix C.2.

Result. Figure 3 shows a significant increase in the proportion of reflective responses across Qwen models, even with only 8k long CoT training examples. Specifically, the reflection ratio increases sharply from below 5% to approximately 75%, indicating that reflective behaviors are quickly picked up and internalized. Additionally, Figure 4 underscores that reflective responses are substantially longer than non-reflective ones. Notably, for both the 1.5B and 3B models, reflective responses consistently exceed non-reflective ones by approximately 2,000 tokens, a trend that holds across all training data scales. These findings suggest that the acquisition of reflective behavior is closely tied to the growth in response length. These findings support Hypothesis 1: with limited long CoT supervision, models quickly adopt surface features of system-2 reasoning, especially reflection which contributes to longer responses.

3.3 Cumulative Error Analysis

While the previous analysis focused on the emergence of surface-level reasoning patterns, it did not directly assess how output length impacts answer accuracy. Hypothesis 2 posits that longer responses increase the risk of cumulative errors, thereby reducing overall accuracy. In this subsection, we test this hypothesis in a controlled setting.

Standard mathematical benchmarks introduce confounding factors such as domain knowledge, problem interpretation, and strategy selection. These complexities make it difficult to isolate the effect of response length on performance. To address this, we design a synthetic arithmetic benchmark that controls for external variables while preserving step-by-step reasoning structure. This allows us to directly examine how output length correlates with error accumulation.

Benchmark Design. Each instance in our synthetic dataset is a randomly generated arithmetic expression composed of 5 to 15 operations. To ensure controlled difficulty and interpretability:

- Operands are uniformly sampled from the range $[1, 100]$.
- Operators include addition, subtraction, multiplication, and division.
- All intermediate results are constrained to be integers.
- Each problem requires a fixed number of simple, sequential reasoning steps.

This setup enables a rigorous empirical evaluation of Hypothesis 2 by explicitly correlating response length (number of arithmetic steps) with accuracy under consistent and controlled conditions. An example problem from the benchmark is shown in Figure 6.

Setup. We evaluate the models described in Section 2.2 on 400 synthetic arithmetic problems. Fol-

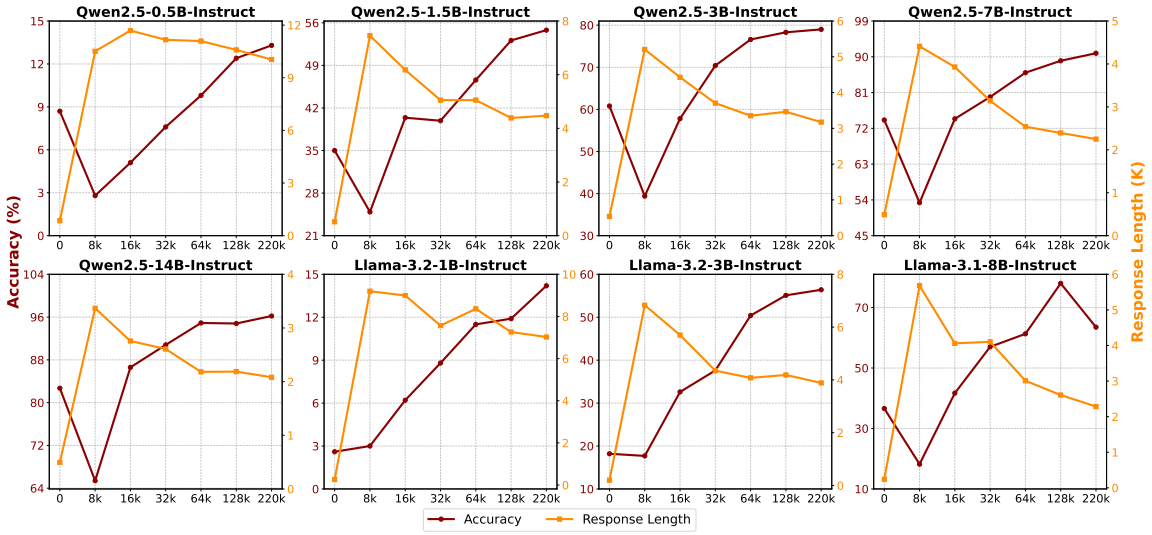

Figure 5: Arithmetic accuracy and response length on our synthetic benchmark for models trained with increasing amounts of long CoT data. Most models exhibit a sharp drop in arithmetic accuracy and a corresponding increase in response length after training on the 8k subset, with the exception of Llama-3.2-{1B,3B}-Instruct, whose initial performance is already low (<20% accuracy).

lowing the same evaluation protocol as in Section 2.2, we use sampling with temperature 0.6 and top-p 0.95 to generate $k = 4$ responses per problem. We report the average accuracy across the k responses (avg@k).

Result. As shown in Figure 5, most models experience a significant drop in arithmetic accuracy after training on the 8k long CoT subset, accompanied by a substantial increase in response length. For instance, Qwen2.5-7B-Instruct exhibits a 30% drop in accuracy, while its average output length grows from approximately 600 to 3,600 tokens. With more long CoT data, performance gradually recovers. These trends closely mirrors the degradation and recovery patterns observed on real-world math benchmarks (Fig. 1 and 2). Qualitative analysis further supports these findings. As shown in Appendix B.5, the model trained on 8k CoT data frequently generates verbose responses with repetitive phrasing (e.g., "wait") and restates equations multiple times, yet still makes similar arithmetic mistakes that compound across steps despite proposing alternative solutions. In contrast, the model trained on 64k CoT data demonstrates more disciplined reasoning behavior. It verifies steps more effectively and proposes alternatives with clearer intent.

Overall, these results offer strong empirical support for Hypotheses 2: although longer responses may enable more elaborate reasoning, they also increase the likelihood of compounding errors, resulting in *Long CoT Degradation*.

4 Impact of Long CoT Supervision on RL

As we continue exploring the impact of different methods for using long CoT data to train SLMs, the next natural research question for us to investigate is how the performance of these SLMs, after training with long CoTs under different settings, might affect the subsequent RL stage.

Existing approaches generally fall into two categories: 1) using a limited number of long CoT examples (≤10k) during SFT as a cold start for subsequent RL (Guo et al., 2025; Ren et al., 2025), and 2) applying RL to models heavily distilled with substantial amounts of long CoT data (>100k) (Luo et al., 2025; Michael Luo, 2025; Liu et al., 2024). However, a systematic understanding of how varying levels of long CoT supervision influence subsequent RL performance remains underexplored – particularly in the context of SLMs.

To address this limitation, we systematically analyze how different amounts of long CoT supervision during SFT impact RL outcomes, particularly focusing on performance degradation, token efficiency, and overall learning dynamics.

4.1 Training Setup

We conduct RL training on three models, Qwen2.5-{0.5B, 1B, 3B}-Instruct, which represent small language models with distinct degradation patterns observed in prior SFT experiments (§2).

For training, we adopt the dataset configuration introduced by Zeng et al. (2025), where the

Question:
Compute the answer to this arithmetic expression: 93/3+77+(12-32)-11+61*78+44*39+55

Ground_Truth: 6606

Model's Answer (Before Training):
To compute the arithmetic expression, we will follow the order of operations (PEMDAS/BODMAS):

1. Parentheses/Brackets: Solve the expression inside the parentheses first. 12 - 32 = -20

.....

So, the final answer is: **6606**

Number of Calculation: 10 **Response Length: 502**

Model's Answer (After Training):
Okay, let's see. I need to compute this arithmetic expression:

.....

Let me compute the subtraction and addition from left to right. Starting with 108 - 20. That's 88. Then, 88 - 11 is 77. Then, 77 + 4758. Wait, 77 plus 4758. Let me add those. 70 plus 4750 is 4820, and 7 plus 58 is 65, so 4820 + 65 is 4885. So after the subtraction and addition so far, we have 4885.

.....

So the final answer is 6601. Let me double-check my steps to make sure I didn't make a mistake.

First, 93 divided by 3 is 31. Then, 31 + 77 is 108. Then, 12 - 32 is -20. So the expression becomes 108 -20 -11 + 4758 +1716 +55. Then, 108 -20 is 88, 88 -11 is 77. Then, 77 +4758 is 4885, 4885 +1716 is 6601. Yeah, that seems right. Let me verify each multiplication again.

.....

Therefore, the answer is **6601**

Number of Calculation: 40 **Response Length: 1405**

Figure 6: A sample problem from our synthetic arithmetic benchmark, with answers from Qwen2.5-1.5B-Instruct before and after training on 8k long CoT examples. "Number of Calculation" indicates the total number of arithmetic operations performed in the response. Errors in the model's intermediate reasoning are highlighted in red.

datasets used for each model have been empirically shown to be of moderate difficulty and effective for their respective scales. The 0.5B model is trained on the *Medium* difficulty subset, composed of MATH level 1-4 problems (Hendrycks et al., 2021), while the 1B and 3B models are trained on the *Hard* subset, which includes MATH level 3-5 problems (Hendrycks et al., 2021). Both subsets contain approximately 8,000 examples.

We perform RL training using the verl (Sheng et al., 2024) framework. We employ the GRPO (Shao et al., 2024) algorithm with a rule-based binary reward function: the model receives a reward of 1 if the final answer is correct, and 0 otherwise. The training uses a prompt batch size of 1,024, generates 8 rollouts per prompt, and sets the maximum rollout length to 8,192 tokens. Optimization is performed using a mini-batch size of 256. All models are trained with the same set of hyperparameters and the evaluation settings remain consistent with those outlined in Section 2.2.

4.2 Results

Long CoT degradation negatively impacts subsequent RL training. Figure 7 shows that starting RL training from a checkpoint fine-tuned with 8k long CoT data consistently yields lower accuracy and longer responses throughout training, as compared to the baseline approach without any SFT. Moreover, the performance gap relative to the baseline persists. These results suggest that the degradation induced by long CoT can hardly be mitigated by RL and continues to adversely affect RL-based optimization.

RL further improves token efficiency in long CoT distilled SLMs. As shown in Figure 7, SLMs fine-tuned with long CoT data (at 0.5B, 1.5B, and 3B scales) exhibit a rapid reduction in response length during the early stages of RL training, accompanied by a steady increase in accuracy. This observation aligns with previous findings (Luo et al., 2025; Liu et al., 2024). In contrast, instruction-tuned baseline models without long CoT SFT show only slight increases in response length and minor accuracy gains under RL, highlighting their limited potential for further improvement (Zeng et al., 2025). These results demonstrate that RL can further enhance the token efficiency of SLMs distilled with long CoT data. Notably, as observed in Section 2, increasing the scale of long CoT SFT data alone also led to continuous improvements in token efficiency, and this trend is further amplified by RL. This finding underscores the synergistic roles of long CoT SFT and RL in maximizing the efficiency of SLMs.

Extensive long CoT SFT training substantially enhances both the efficiency and performance ceiling of subsequent RL. As illustrated in Figure 7, models without prior long CoT SFT show marginal improvement during RL at scales of 0.5B, 1.5B, and 3B. In contrast, initializing RL from a checkpoint trained on 128k long CoT examples results in faster accuracy improvements and a markedly higher final performance. In particular, it is interesting to observe that for the 0.5B model, although long CoT SFT initially results in lower performance compared to the baseline, RL training rapidly closes this gap and further enhances model capabilities. Notably, after RL, the model achieves a 13% improvement over the baseline, and over a 60% gain relative to its pre-RL state. These results indicate that for SLMs, even when long CoT SFT

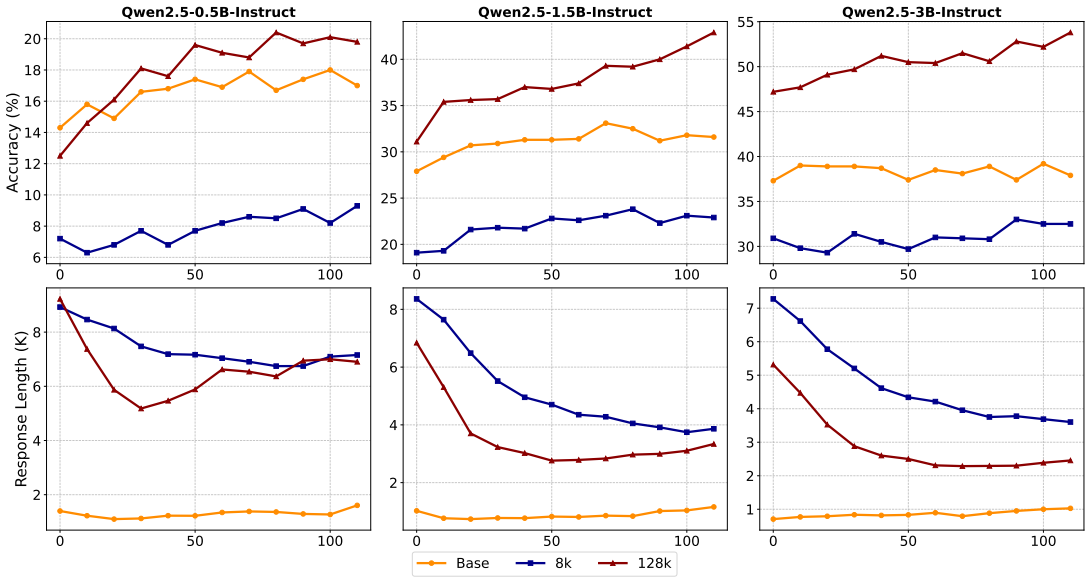

Figure 7: Impact of long CoT SFT data on downstream RL training across Qwen2.5 models. Top: Accuracy of RL-trained models over training steps. Bottom: Average response length during training. Each column corresponds to a different model scale (0.5B, 1.5B, 3B). Each curve represents an SFT data setting: Base (no SFT, serving as a baseline), 8k, and 128k (denoting the number of long CoT examples used during SFT). The horizontal axis in all plots indicates the RL training steps.

temporarily degrades post-SFT performance, large-scale exposure to long CoT data yields substantial benefits during RL training.

Collectively, the results suggest that special care is needed when using long CoT data to improve the reasoning capabilities of small language models. Effective use of such data first requires sufficient exposure during the SFT stage. While the SFT-trained model alone may not achieve strong performance, the subsequent RL stage can substantially enhance its capabilities.

5 Related Work

Recent advances in long CoT reasoning (Jaech et al., 2024; Guo et al., 2025) have substantially improved downstream performance, particularly for large language models (Xiang et al., 2025; Yu et al., 2024; Li et al., 2025b; Huang et al., 2024). While some progress has been made in efficiently distilling long CoT knowledge into smaller models (Muennighoff et al., 2025; Ye et al., 2025; Yeo et al., 2025; Li et al., 2025a), systematic and comprehensive evaluations remain limited.

In parallel, RL for reasoning (Guo et al., 2025; Schulman et al., 2017; Shao et al., 2024; Liu et al., 2025) has achieved promising results across a range of math and coding tasks, including for SLMs (Zeng et al., 2025; Xie et al., 2025; Hu et al., 2025; Bercovich et al., 2025). However, there has been little systematic study on the joint impact of long CoT SFT and RL training on SLMs. This gap motivates our present study[6].

6 Conclusion

In this work, we conduct a systematic study of how the scale of long CoT data impacts small language models (SLMs). Our findings reveal a consistent phenomenon, *Long CoT Degradation*, where limited long CoT supervision significantly impairs model performance across diverse model families and sizes. Through comprehensive experiments, we further analyze the mechanisms behind this degradation, attributing it to error accumulation induced by excessively verbose outputs – an issue to which SLMs are particularly vulnerable. Finally, we show that while long CoT SFT can hinder subsequent RL when data is insufficient, scaling up long CoT supervision enables RL to achieve both greater efficiency and higher final performance.

Our work highlights the pitfalls and misconceptions in current long CoT usage for SLM training, identifies key limitations, and offers practical guidance for addressing them. We also hope this work sheds light on future research aimed at designing optimal pipelines for building effective reasoning models.

[6]Due to space limitations, a comprehensive review is provided in Appendix A.

Limitations

While our work offers new insights into the effects of long CoT for SLMs, we acknowledge several limitations. First, our analysis focuses on mathematical reasoning, which provides a well-structured setting and reliable evaluation metrics for studying long CoT supervision. While the findings may offer insights applicable to domains such as logical reasoning or code generation, we do not directly study these areas. Extending our framework to these areas is a promising direction for future work, though acquiring large-scale, high-quality long CoT training data in such domains remains a practical challenge. Second, although we study multiple model families and sizes and observe consistent patterns across them, we do not explicitly isolate the impact of pre-training data composition. Prior work suggests that pre-training plays an important role in shaping long CoT reasoning capabilities. A more controlled investigation into how pre-training interacts with long CoT supervision, particularly for small models, would provide valuable complementary insights.

Ethical Statement

This study investigates the effects of long chain-of-thought supervision on small language models using publicly available models and datasets. The research does not involve human subjects or any sensitive or proprietary data. This work does not propose or support any applications with foreseeable potential for harm or misuse. In experiments, we comply with all licenses for models, data and code.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. 2024. On-policy distillation of language models: Learning from self-generated mistakes. In *Proceedings of ICLR*.

Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, and 1 others. 2025. Llama-nemotron: Efficient reasoning models. *arXiv preprint arXiv:2505.00949*.

Zhipeng Chen, Yingqian Min, Beichen Zhang, Jie Chen, Jinhao Jiang, Daixuan Cheng, Wayne Xin Zhao, Zheng Liu, Xu Miao, Yang Lu, and 1 others. 2025. An empirical study on eliciting and improving r1-like reasoning models. *arXiv preprint arXiv:2503.04548*.

Hugging Face. 2025. Open r1: A fully open reproduction of deepseek-r1.

Tao Feng, Yicheng Li, Li Chenglin, Hao Chen, Fei Yu, and Yin Zhang. 2024. Teaching small language models reasoning through counterfactual distillation. In *Proceedings of EMNLP*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of NeurIPS Datasets and Benchmarks Track*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. 2025. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*.

Zhen Huang, Haoyang Zou, Xuefeng Li, Yixiu Liu, Yuxiang Zheng, Ethan Chern, Shijie Xia, Yiwei Qin, Weizhe Yuan, and Pengfei Liu. 2024. O1 replication journey–part 2: Surpassing o1-preview through simple distillation, big progress or bitter lesson? *arXiv preprint arXiv:2411.16489*.

Tingfeng Hui, Lulu Zhao, Guanting Dong, Yaqi Zhang, Hua Zhou, and Sen Su. 2024. Smaller language models are better instruction evolvers. *arXiv preprint arXiv:2412.11231*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Hojae Lee, Junho Kim, and SangKeun Lee. 2024. Mentor-kd: Making small language models better multi-step reasoners. In *Proceedings of EMNLP*.

Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. 2024. Numinamath.

Liunian Harold Li, Jack Hessel, Youngjae Yu, Xiang Ren, Kai-Wei Chang, and Yejin Choi. 2023. Symbolic chain-of-thought distillation: Small models can also "think" step-by-step. In *Proceedings of ACL*.

Yuetai Li, Xiang Yue, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, Bhaskar Ramasubramanian, and Radha Poovendran. 2025a. Small models struggle to learn from strong reasoners. *arXiv preprint arXiv:2502.12143*.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, and 1 others. 2025b. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*.

Zihan Liu, Yang Chen, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. Acemath: Advancing frontier math reasoning with post-training and reward modeling. *arXiv preprint arXiv:2412.15084*.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. Notion Blog.

Roy Huang Ameen Patel Alpay Ariyak Qingyang Wu Xiaoxiang Shi Rachel Xin Colin Cai Maurice Weber Ce Zhang Li Erran Li Raluca Ada Popa Ion Stoica Michael Luo, Sijun Tan. 2025. Deepcoder: A fully open-source 14b coder at o3-mini level. Notion Blog.

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. *arXiv preprint arXiv:2501.19393*.

OpenAI. 2025. Openai o3 and o4-mini system card.

ZZ Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, and 1 others. 2025. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*.

Gemma Team. 2025. Gemma 3.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

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

Liang Wen, Yunke Cai, Fenrui Xiao, Xin He, Qi An, Zhenyu Duan, Yimin Du, Junchen Liu, Lifu Tang, Xiaowei Lv, and 1 others. Light-r1: Curriculum sft, dpo and rl for long cot from scratch and beyond. *arXiv preprint arXiv:2503.10460*.

Yuyang Wu, Yifei Wang, Tianqi Du, Stefanie Jegelka, and Yisen Wang. 2025. When more is less: Understanding chain-of-thought length in llms. *arXiv preprint arXiv:2502.07266*.

Violet Xiang, Charlie Snell, Kanishk Gandhi, Alon Albalak, Anikait Singh, Chase Blagden, Duy Phung, Rafael Rafailov, Nathan Lile, Dakota Mahan, and 1 others. 2025. Towards system 2 reasoning in llms: Learning how to think with meta chain-of-though. *arXiv preprint arXiv:2501.04682*.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. 2025. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*.

Haotian Xu, Xing Wu, Weinong Wang, Zhongzhi Li, Da Zheng, Boyuan Chen, Yi Hu, Shijia Kang, Jiaming Ji, Yingying Zhang, and 1 others. 2025. Redstar: Does scaling long-cot data unlock better slow-reasoning systems? *arXiv preprint arXiv:2501.11284*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*.

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. 2025. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*.

Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. 2024. Distilling system 2 into system 1. *arXiv preprint arXiv:2407.06023*.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. 2025. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. 2024. LlamaFactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of ACL*.

A Related Work

A.1 Long Chain-of-Thought

The paradigm of long CoT reasoning is first popularized by OpenAI-o1 (Jaech et al., 2024), and becomes widely accessible with the open-sourcing of DeepSeek-R1 (Guo et al., 2025). Outputs from these LRMs are characterized by extended and reflective CoT sequences, which not only exhibit behaviors of system-2 cognition (Xiang et al., 2025; Yu et al., 2024; Li et al., 2025b), but also provide richer intermediate supervision signals for knowledge distillation (Hinton et al., 2015). As a result, downstream model performance is substantially improved (Huang et al., 2024).

Subsequent research focus on improving the efficiency of long CoT distillation. Notably, recent studies (Muennighoff et al., 2025; Ye et al., 2025) have identified compact yet highly effective prompt subsets, demonstrating that strong performance can be achieved with as few as 1,000 training examples. However, these efforts have predominantly concentrated on large-scale models (e.g., those with 32 billion parameters), and the impact of long chain-of-thought reasoning on smaller models remains largely unexplored.

Preliminary investigations into small language models (8B parameters and below) include (Yeo et al., 2025; Li et al., 2025a). While these studies provide valuable insights into long CoT SFT for small language models, they are based on relatively limited datasets generated by the QwQ-Preview model, which was developed before the widespread adoption of zero-RL training (Team, 2024). This limitation raises concerns about the generalizability of their findings to contemporary training paradigms.

A.2 RL for Reasoning

The release of DeepSeek-R1 (Guo et al., 2025) has sparked broad interest in the RL for reasoning training paradigm, which involves applying reinforcement learning directly to base models using rule-based rewards in conjunction with established online RL algorithms (Schulman et al., 2017; Shao et al., 2024; Liu et al., 2025).

RL has proven highly effective on reasoning tasks with easily verifiable reward signals, particularly in math and coding (Xie et al., 2025; Hu et al., 2025; Bercovich et al., 2025). Notably, recent findings suggest that RL remains effective even for small-scale language models, underscoring its broad applicability (Zeng et al., 2025).

While prior work has shown that initial training with short-CoT SFT can constrain the benefits of subsequent RL (Zeng et al., 2025) in reasoning tasks, recent studies combining extensive Long CoT SFT with RL report superior performance, sometimes exceeding that of larger models trained via alternative pipelines (Luo et al., 2025; Liu et al., 2024; Michael Luo, 2025). However, systematic evaluations of long CoT SFT followed by RL, particularly in the context of small language models, remain limited. This gap motivates the present study's comprehensive investigation.

B Detailed Experimental Setups

B.1 Models

Category	Models
Qwen Family	Qwen2.5-0.5B-Instruct, Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, Qwen2.5-14B-Instruct
Llama Family	Llama3.2-1B-Instruct, Llama3.2-3B-Instruct, Llama3.1-8B-Instruct
Gemma Family	Gemma3-1B-IT

Table 1: Overview of models investigated in this work.

Table 1 summarizes all models evaluated in this study.

B.2 Supervised Fine-Tuning

Supervised Fine-Tuning (SFT) is conducted on a dataset $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$, where each prompt $x^{(i)}$ is paired with an output $y^{(i)}$, which may include a long CoT. The objective is to maximize

the conditional log-likelihood $\log p_\theta(y^{(i)} \mid x^{(i)})$, encouraging the model to reproduce high-quality responses with structured reasoning.

Hyper-parameter	Long CoT Data Amount					
	8k	16k	32k	64k	128k	220k
Number of Epochs	4	4	4	3	3	2
Batch Size	8	16	32	64	128	128
Learning Rate	5×10^{-5}					
Optimizer	Adamw					
Learning Rate Scheduler	cosine					
Max Sequence Length	16384					
Warmup ratio	0.05					
Training Precision	bfloat16					

Table 2: Hyperparameters used for full-parameter supervised fine-tuning.

Training Setup. Our SFT training is conducted using LLaMA-Factory (Zheng et al., 2024) on a server equipped with 8 H100 (80GB) GPUs. The SFT experiments consumed approximately 2,500 GPU hours in total. We adopt full-parameter fine-tuning for all SFT experiments. The detailed hyper-parameters we used are presented in Table 2, which are determined through a preliminary hyperparameter search.

B.3 Reinforcement Learning

DeepSeek-R1 (Guo et al., 2025) conducts large-scale RL using long CoT supervised fine-tuning as a cold start, establishing a widely adopted training pipeline for reasoning-oriented models. In this work, we adopt the same setup to study how long CoT supervision during SFT influences model behavior in the subsequent RL stage.

For the RL algorithm, we use GRPO (Shao et al., 2024), a computationally efficient variant of PPO (Schulman et al., 2017) that eliminates the need for a separate value model by estimating advantages using group-normalized rewards.

In line with DeepSeek-R1 and similar works, we employ a rule-based binary reward function: the model receives a reward of 1 if the final answer is correct, and 0 otherwise. This simple yet effective setup allows us to isolate the effect of long CoT SFT on the optimization behavior during RL.

Training Setup. Our RL training is conducted using verl (Sheng et al., 2024) framework, on a server equipped with eight H100 GPUs (each with 80GB of memory). The RL experiments consumed approximately 5,000 GPU hours in total. The detailed hyperparameters used in our experiments are shown in Table 3.

Hyper-parameter	Value
Training Algorithm	GRPO
Prompt Batch Size	1024
Rollout Per Prompt	8
Maximum Rollout Length	8192
Mini-Batch Size	256
Sampling Temperature	1.0
KL Loss Coefficient	1×10^{-4}

Table 3: Hyperparameters used for RL training.

B.4 Reflection Behavior Analysis Setup

We adopt a cross-validation approach, following the methodology proposed in (Liu et al., 2025), combining both keyword detection and LLM-based identification to detect the self-reflective behaviors in model outputs. In particular, the keyword pool used in this work is limited to: recheck, rethink, reassess, reevaluate, re-evaluate, reevaluation, re-examine, reexamine, reconsider, reanalyze, double-check, check again, think again, verify again, and go over the steps.

Figure 9 presents the prompt employed to determine whether a response contains self-reflection behaviors with an LLM-based approach.

B.5 Arithmetic Benchmark

Qualitative Examples. To complement our quantitative analysis, we present three representative examples in Figures 13, 14, and 15. These responses are generated by Qwen2.5-3B-Instruct models under different training settings: the baseline model, the model finetuned on 8k-length CoT data, and the model finetuned on 64k-length CoT data, respectively. The baseline model solves the problem correctly with a clear and concise reasoning process. In contrast, the 8k-finetuned model produces a much longer and more verbose response. Despite proposing multiple alternative solutions, it repeatedly makes the same arithmetic mistakes, which propagate across steps and compound the final error. The response also exhibits disfluency, including repetitive phrases like "wait" and redundant equation restatements. The 64k-finetuned model shows significant improvement: it maintains more coherent structure, reflects on its steps more effectively, and proposes alternatives in a way that leads to the correct solution. These examples reinforce our hypothesis that longer responses introduce more room for error and noise—unless offset

by sufficient training on long-form reasoning patterns.

C Detailed Evaluation Results

C.1 Long CoT Degradation

Figure 11 and 12 show more detailed evaluation results for the experiments in Section 2.

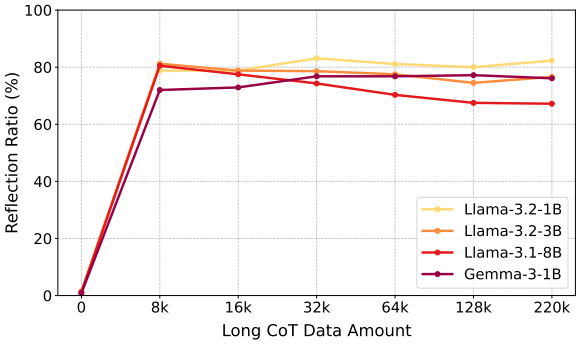

Figure 8: Reflection ratios of LLaMA and Gemma models of different sizes trained on varying amounts of long CoT data. The reflection ratio refers to the proportion of model responses (out of 4,000 on the MATH500 benchmark) that exhibit reflective behavior, as identified through cross-validation.

C.2 Reflection Behavior Analysis

Figure 8 and 10 are the detailed evaluation results for the experiments in Section 3.2.

I will send you a mathematical question along with a detailed response. Your task is to determine whether the response is attempting to answer the question. If the response is off-topic, hallucinated, random talk, or otherwise irrelevant, mark it as 0. Otherwise, assess whether the response exhibits self-reflection.

Categorization Rules:

1. **Category 0**: The response is **off-topic, nonsensical, incoherent, overly repetitive, or lacks logical reasoning.**
   - Example cases:
     - The response does not relate to the question.
     - It contains meaningless or hallucinated content.
     - It consists of excessive repetition without coherence.
2. **Category 1**: The response **attempts to answer the question** but does **not** exhibit self-reflection.
   - Example cases:
     - The response directly solves the problem without revisiting steps.
     - No attempt is made to verify the correctness of the answer or explore alternative solutions.
3. **Category 2**: The response **demonstrates self-reflection** at any level.
   - This may include:
     - **Explicit self-reflection keywords**, such as: *recheck, rethink, reassess, reevaluate, re-evaluate, reevaluation, re-examine, reexamine, reconsider, reanalyze, double-check, check again, think again, verify again, go over the steps*, etc.
     - **Implicit self-reflection behaviors**, such as revisiting the solution, questioning assumptions, or considering alternative approaches **without explicit keywords**.
   - If any form of self-reflection is present, **always categorize it as 2**, regardless of correctness or answer quality.
4. **Category 3**: The response consists **solely of Python code for calculations** without exhibiting self-reflection.
   - Example cases:
     - The response only provides a Python script to compute the solution **without any verification, re-evaluation, or alternative considerations.**

Output Format:

Your response should first provide a **very brief explanation** of your analysis, followed by a **single category number (0, 1, 2, or 3)** at the end. You must include the category number at the end of your response.

Example outputs:

- 'The response is off-topic and does not attempt to answer the question. 0.'
- 'The response provides a direct solution without self-reflection. 1.'
- 'The response demonstrates self-reflection. 2.'
- 'The response consists solely of Python code without any self-reflection. 3.'

Question: {question}

Response: {response}

Figure 9: Prompt template used by GPT-4o-mini for reflection behavior identification.

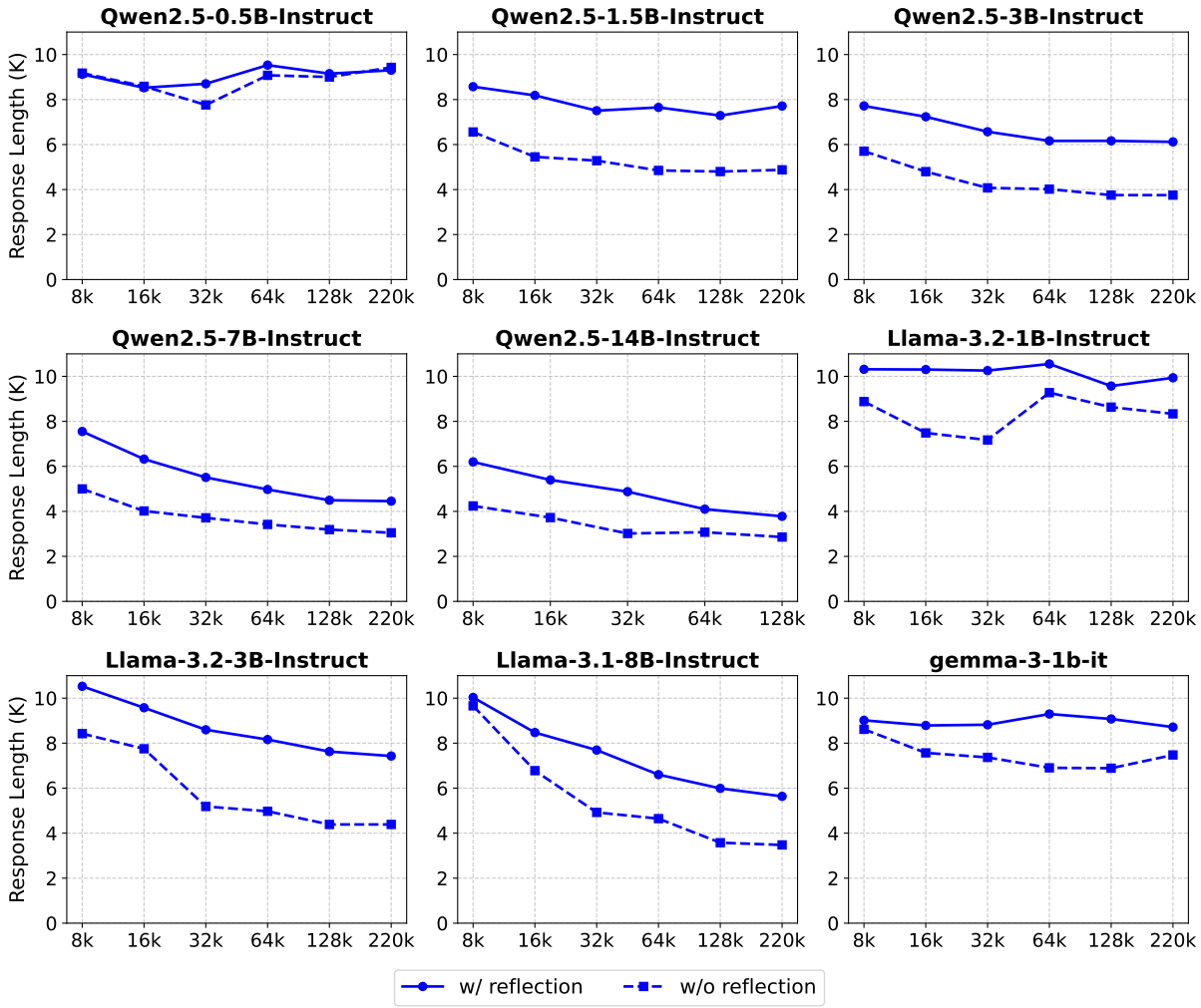

Figure 10: Average response lengths of multiple models trained with varying amounts of long CoT data. Solid lines represent responses exhibiting reflection behavior; dashed lines denote responses without reflection.

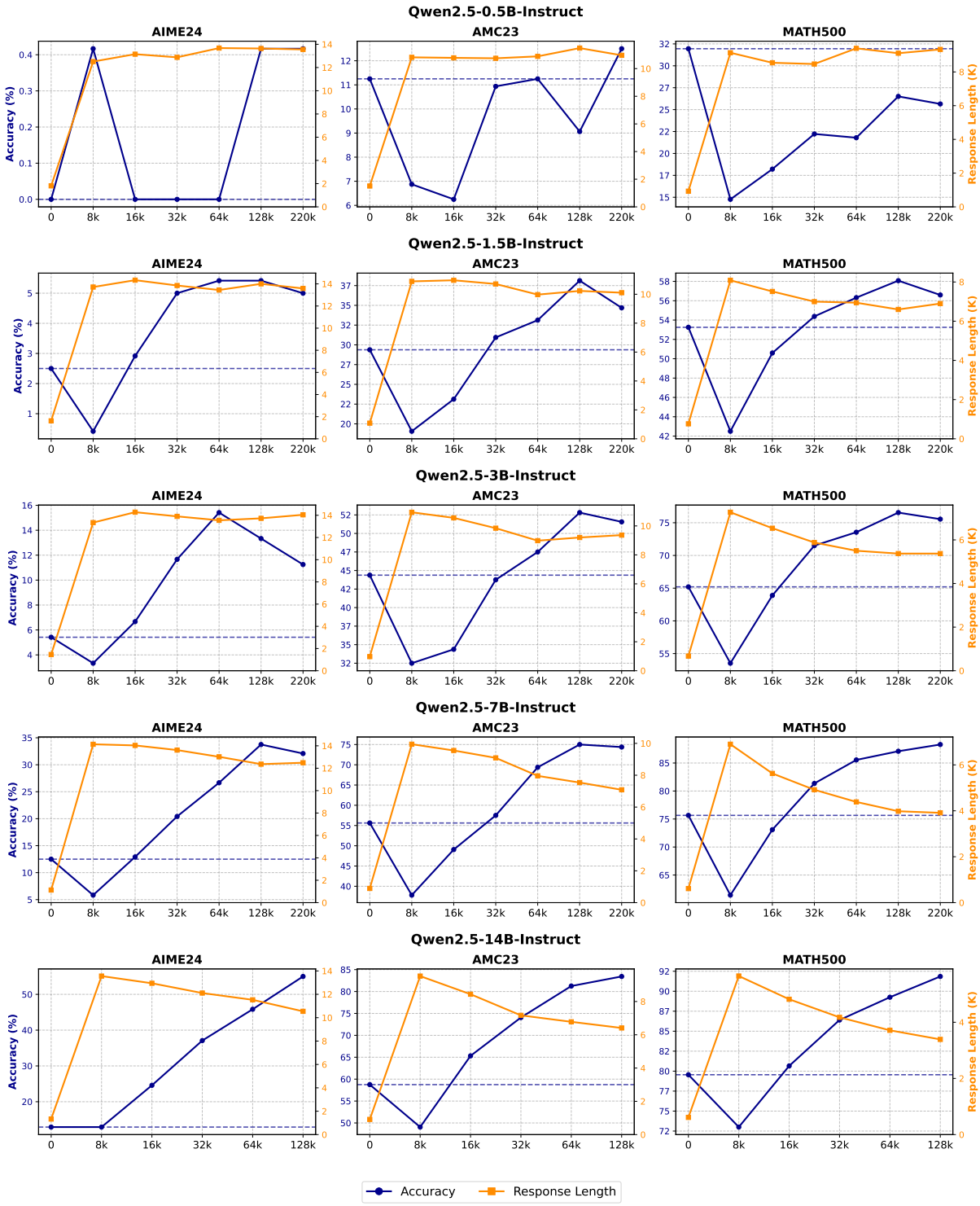

Figure 11: Comprehensive evaluation of Qwen models trained with varying amounts of long CoT data.

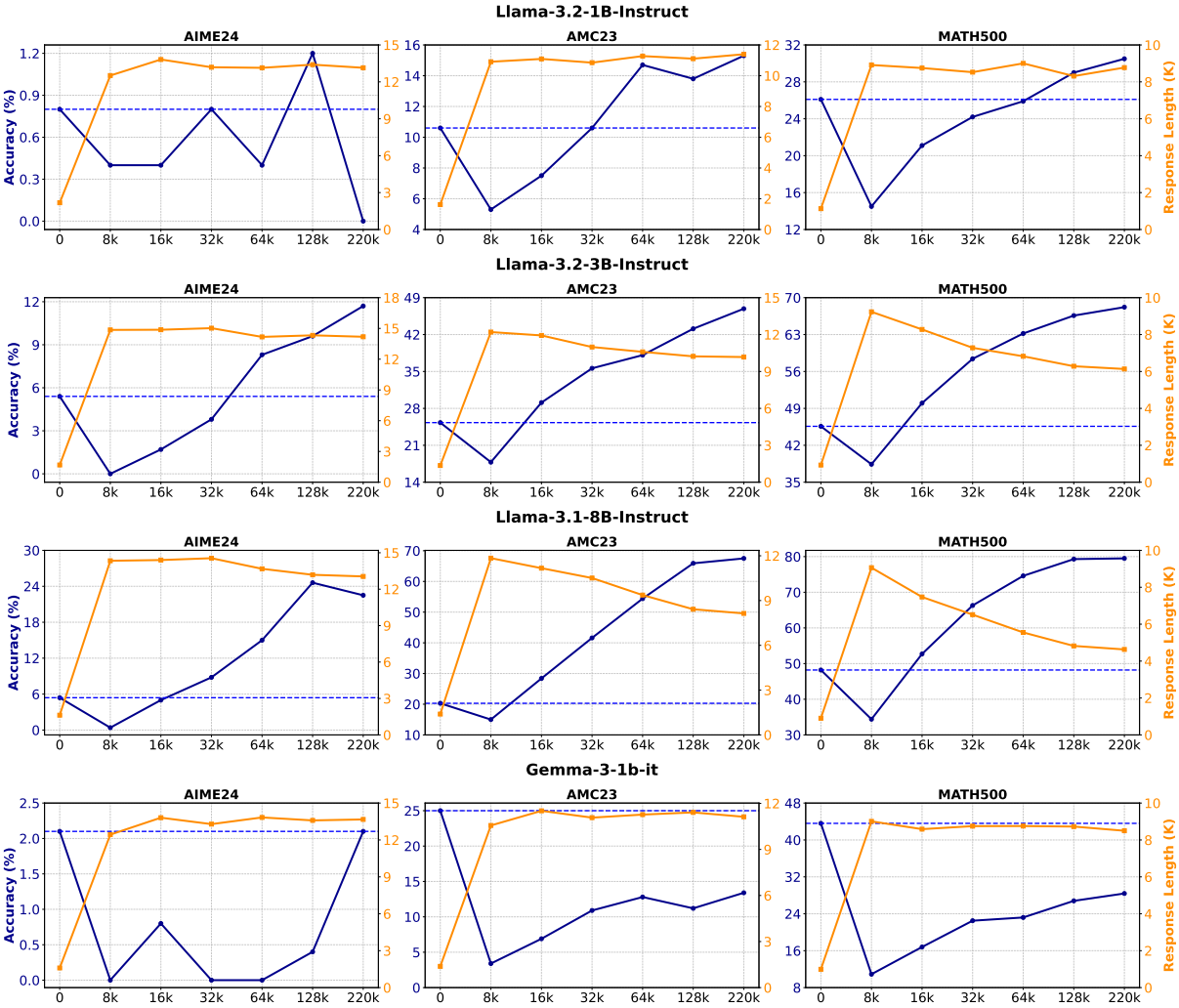

Figure 12: Comprehensive evaluation of LLaMA and Gemma models trained with varying amounts of long CoT data.

Example 1 Qwen2.5-3B-Instruct

Question

Compute the answer to this arithmetic expression: (14*21*38+32+23+31+3)-11+59+71+60

Response

To solve the arithmetic expression $(14 \times 21 \times 38 + 32 + 23 + 31 + 3) - 11 + 59 + 71 + 60$, we need to follow the order of operations, which means we should perform the multiplication before the addition and subtraction.
First, calculate the product $14 \times 21 \times 38$:

$$14 \times 21 = 294$$

$$294 \times 38 = 11172$$

Now substitute this back into the expression:

$$11172 + 32 + 23 + 31 + 3 - 11 + 59 + 71 + 60$$

Next, add the numbers inside the parentheses:

$$11172 + 32 + 23 + 31 + 3 = 11261$$

Now, substitute this back into the expression:

$$11261 - 11 + 59 + 71 + 60$$

Perform the subtraction and addition from left to right:

$$11261 - 11 = 11250$$

$$11250 + 59 = 11309$$

$$11309 + 71 = 11380$$

$$11380 + 60 = 11440$$

Thus, the final answer is:

$$\boxed{11440}$$

Figure 13: Example from our synthetic arithmetic benchmark. The response is generated by the Qwen2.5-3B-Instruct model. The model follows a step-by-step approach, applying the order of operations to solve the arithmetic expression.

Example 2 Qwen2.5-3B-Instruct 8k

Question

Compute the answer to this arithmetic expression: (14*21*38+32+23+31+3)-11+59+71+60

Response

Okay, let's see. I need to compute this arithmetic expression: (14*21*38 + 32 + 23 + 31 + 3) - 11 + 59 + 71 + 60. Hmm, that looks a bit complicated, but let's break it down step by step.
First, I should handle the operations inside the parentheses before dealing with the numbers outside. The expression starts with 14 multiplied by 21 multiplied by 38. Let me calculate that first. Multiplying those numbers might take some time, so maybe I can do them step by step. Let's see, 14 times 21. Hmm, 14*20 is 280, and 14*1 is 14, so 280+14=294. Then, multiplying that result by 38. Okay, 294*38. Let me break that down too. Maybe 294*30 is 8820, and 294*8 is 2352. Adding those together, 8820+2352. Let me add them step by step: 8820 + 2000 is 10820, then +352 more is 11172. So, 14*21*38 equals 11172.
Next, the problem adds 32 to that result. So, 11172 + 32. That's straightforward, 11172 + 30 is 11202, plus 2 more is 11204. Then, it adds 23: 11204 +23. Let's see, 11204 +20 is 11224, plus 3 is 11227. Next, adding 31: 11227 +31. 11227 +30 is 11257, plus 1 more is 11258. Then, adding 3: 11258 +3. That's just 11261. So, inside the parentheses simplifies to 11261.
Now, the expression continues with minus 11: 11261 -11. That's easy, 11261 -10 is 11251, minus 1 more is 11250. Then, adding 59: 11250 +59. Let me check that. 11250 +50 is 11290, plus 9 is 11299. Next, adding 71: 11299 +70 is 11369, plus 1 is 11370. Then, adding 60: 11370 +60. That's 11370 +60. Let me verify. 11370 +60: 11370 +50 is 11320, plus 10 is 11330, plus 10 is 11340. Wait, no, hold on. Wait, 11370 +60: 11370 +60. Let me do this differently. 11370 + 60 is the same as 11370 + 6*10. So, 11370 +60 = 11370 + 6*10. 11370 +60 is 11370 + 60. Alternatively, 11370 + 60 = 11370 + 60. Let me add 11370 + 60. 11370 + 60: 11370 + 50 is 11320, then +10 is 11330. Wait, that doesn't seem right. Wait, 11370 +60: 11370 + 60. Let's think of 11370 as 1137*10. Then 1137*10 +6*10 = 11370 +60. Alternatively, 11370 +60: 11370 + 60. Let's do 11370 + 60. 11370 + 50 = 11320, then +10 = 11330. Wait, but 11370 +60 is actually 11370 + 6*10. So, 11370 +60 = 11370 + 60. Let me verify with another approach. 11370 +60: 11370 + 60. If I break down 60 into 50 +10, then 11370 +50 = 11320, then +10 = 11330. Alternatively, 11370 +60: 11370 + 60. Let me do 11370 + 60. Let's count from 11370: adding 10 is 11380, adding another 10 is 11390. Wait, no. Wait, 11370 +60: 11370 + 60. Let me use column addition.
.....
Wait, that's correct. So, 11370 +60 = 11430. So, in that case, 11370 +60 = 11430. So, 11430 is the sum of 11370 and 60. Okay, so that's correct. Therefore, 11261 -11 +59 +71 +60 equals 11261 -11 + (59+71+60). Let me compute 59+71 first. 59+71: 50+70 is 120, 9+1 is 10, so 120+10=130. Then, 130+60 is 190. So, 59+71+60=190. Then, 11261 -11 is 11250, plus 190 is 11250 +190. Let me add those. 11250 +190: 11250 +100 is 11350, then +90 is 11350 +90=11340. So, the total is 11340.
Wait, let me check that again. 11261 -11 is 11261 -10=11251, minus 1=11250. Then, 11250 +59=11250+59. Let's do 11250 +50=11290, then +9=11299. Then, 11299 +71=11299+70=11369, +1=11370. Then, 11370 +60=11430. So, total is 11430.
Alternatively, maybe there's a different way to compute the total without breaking it down so many times. Let me see. The original expression is (14*21*38 +32+23+31+3) -11+59+71+60. So, the first part is 14*21*38 + (32+23+31+3).
.....
So, that's 11261. Then, 11261 -11=11250, +59=11250+59=11250+50+9=11290+9=11299. Then, +71=11299+70=11369+1=11370. Then, +60=11370+60=11430. So, total is 11430.
.....
Wait, maybe the error is in the initial step of adding the numbers inside the parentheses. Let me check that again. The first part is (14*21*38 +32 +23 +31 +3). Let's compute each term:
14*21*38: 14*21=294, 294*38=11172. Correct.
.....
So, inside the parentheses is 11261. Then, subtract 11: 11261 -11=11250. Correct.
Then, add 59: 11250 +59=11250 +50 +9=11290 +9=11299. Correct.
Add 71: 11299 +71=11299 +70 +1=11369 +1=11370. Correct.
Add 60: 11370 +60=11370 +60=11430. Correct.
So, the total is 11430. Therefore, the first method is correct. Then, why does the second method give a different result? Because in the second method, the user added the constants first. .....
Therefore, the total is 11430. Therefore, the second method is incorrect because it misapplies the order of operations. The user added the constants first, which is not the case in the original expression. The original expression requires the constants to be added after the product is computed and before the subtraction and additions. Therefore, the correct total is 11430.
Therefore, the final answer is $\boxed{11430}$.

Figure 14: Example from our synthetic arithmetic benchmark. The response is generated by Qwen2.5-3B-Instruct finetuned on 8k long CoT data. Despite proposing multiple alternative solutions, the model makes similar calculation errors, which propagate through subsequent steps and result in compounding mistakes. Incorrect steps are highlighted in red. The response is truncated as it exceeds 5,000 tokens.

Example 3 Qwen2.5-3B-Instruct 64k

Question

Compute the answer to this arithmetic expression: (14*21*38+32+23+31+3)-11+59+71+60

Response

Okay, so I need to compute this arithmetic expression: (14*21*38 + 32 + 23 + 31 + 3) - 11 + 59 + 71 + 60. Hmm, let me break this down step by step. First, I should handle the operations inside the parentheses, which is the product of 14, 21, and 38, then add the other numbers. Let me start with that.
Calculating 14 multiplied by 21. Let me do 14*20 first, which is 280, and then add 14*1, which is 14. So 280 + 14 = 294. Okay, so 14*21 is 294. Now, I need to multiply that result by 38. Hmm, 294*38. Let me break that down. Maybe 294*40 would be easier, but subtract 294*2. Wait, 294*40 is 11,760 and 294*2 is 588. So 11,760 - 588 = 11,172. So 14*21*38 is 11,172. Let me check that again to be sure. 294*38: 294*30 is 8,820 and 294*8 is 2,352. Adding those together: 8,820 + 2,352. Let's see, 8,000 + 2,000 is 10,000, 820 + 352 is 1,172. So 10,000 + 1,172 is 11,172. Yep, that matches. So that part is correct.
Now, add the numbers 32, 23, 31, and 3. Let's do that step by step. 32 + 23 is 55. Then 55 + 31 is 86. Then 86 + 3 is 89. So the sum inside the parentheses is 11,172 + 89. Let me compute that. 11,172 + 80 is 11,252, then +9 is 11,261. So the parentheses part equals 11,261.
Now the entire expression is 11,261 - 11 + 59 + 71 + 60. Let me handle the subtraction and addition from left to right. So first, 11,261 - 11. That's straightforward: 11,261 - 10 is 11,251, then subtract 1 more is 11,250. So now we have 11,250 + 59 + 71 + 60.
Let me add those numbers. 59 + 71. 50 + 70 is 120, 9 + 1 is 10, so total 130. Then 130 + 60 is 190. So adding 190 to 11,250 gives 11,250 + 190 = 11,440. Let me verify each step again to make sure I didn't make a mistake.
First, 14*21*38: 14*21 is 294, 294*38 is 11,172. Then adding 32+23+31+3: 32+23=55, 55+31=86, 86+3=89. So 11,172 + 89 = 11,261. Then 11,261 - 11 = 11,250. Then adding 59+71+60: 59+71=130, 130+60=190. 11,250 + 190 = 11,440. Seems correct.
Wait, let me check the addition again. 59 + 71: 59 + 70 is 129, plus 1 is 130. Then 130 + 60 is 190. Correct. Then 11,250 + 190: 11,250 + 100 is 11,350, plus 90 is 11,440. Yes. So the final answer should be 11,440.
But just to be thorough, let me compute the entire expression step by step again, maybe using a different order or verify the multiplication another way.
Alternatively, maybe I can compute 14*21*38 again. Let's see, 14*21 is 294, then 294*38. Let's compute 294*38 as (300 - 6)*38. So 300*38 = 11,400, 6*38=228. So 11,400 - 228 = 11,172. Same result. Then adding 32+23+31+3: 32+23=55, 55+31=86, 86+3=89. So 11,172 + 89 = 11,261. Then subtract 11: 11,261 - 11 = 11,250. Then add 59,71,60. Let's do 59 + 71 first: 59 + 71. 59 + 70 is 129, plus 1 is 130. Then 130 + 60 is 190. So 11,250 + 190 is 11,440. All steps check out.
Alternatively, maybe compute the entire expression as written:
(14*21*38 + 32 + 23 + 31 + 3) - 11 + 59 + 71 + 60
Let me compute all the additions first before subtracting:
First, compute the sum inside the parentheses: 14*21*38 + 32 + 23 + 31 + 3.
But we already did that. Then subtract 11, then add 59, 71, 60.
Alternatively, maybe compute each term step by step:
Start with 14*21*38. Let's compute that again. 14*21 is 294. Then 294*38. Let me compute 294*38 as 294*(30 + 8) = 294*30 + 294*8. 294*30 is 8,820. 294*8 is 2,352. Adding those gives 8,820 + 2,352 = 11,172. Correct.
Then 32 + 23 + 31 + 3. Let's add them in pairs: 32 + 31 = 63, 23 + 3 = 26. Then 63 + 26 = 89. Correct. So the parentheses total 11,172 + 89 = 11,261.
Now subtract 11: 11,261 - 11 = 11,250. Then add 59, 71, 60. Let's add 59 + 71 first. 59 + 70 is 129, plus 1 is 130. Then 130 + 60 = 190. Then 11,250 + 190 = 11,440. All steps confirm the same result.
Alternatively, maybe compute the entire expression as written:
14*21*38 + 32 + 23 + 31 + 3 - 11 + 59 + 71 + 60
But that's the same as before. Alternatively, compute each term in order:
14*21*38 = 11,172
.....
+60 = 11,440
Yes, same result. So regardless of the order, the answer is 11,440.
I think that's solid. Let me just check if I made any arithmetic errors in the final addition. 11,250 + 190. 11,250 + 100 = 11,350. Then +90 is 11,440. Correct.
Alternatively, 190 is 100 + 90. 11,250 + 100 is 11,350. 11,350 + 90 is 11,440. Yep.
So I think that's the correct answer. 11,440. So the final answer is $\boxed{11440}$.
Final Answer $\boxed{11440}$

Figure 15: Example from our synthetic arithmetic benchmark. Response is generated by Qwen2.5-3B-Instruct finetuned on 64k long CoT data. The model frequently engages in verification steps and occasionally proposes alternative solutions. Verification steps and alternative solution paths are highlighted in blue.