

# Innate Reasoning is Not Enough: In-Context Learning Enhances Reasoning Large Language Models with Less Overthinking

Anonymous ACL submission

## Abstract

Recent advances in Large Language Models (LLMs) have introduced Reasoning Large Language Models (RLLMs), which employ extended thinking processes with reflection and self-correction capabilities, demonstrating the effectiveness of test-time scaling. RLLMs exhibit innate Chain-of-Thought (CoT) reasoning capability obtained from training, leading to a natural question: “Is CoT prompting, a popular In-Context Learning (ICL) method for chat LLMs, necessary to enhance the reasoning capability of RLLMs?” In this work, we present the first comprehensive analysis of the impacts of Zero-shot CoT and Few-shot CoT on RLLMs across mathematical reasoning tasks. We examine models ranging from 1.5B to 32B parameters, finding that contrary to concerns, CoT prompting significantly enhances RLLMs’ performance in most scenarios. Our results reveal distinct patterns: large-capacity models show minimal improvement on simple tasks but substantial gains on complex problems, while smaller models exhibit the opposite behavior. Further analysis demonstrates that CoT prompting effectively controls the distribution of the numbers of thinking tokens and reasoning steps, reducing excessive reflections by approximately 90% in some cases. Moreover, attention logits analysis reveals the RLLMs’ overfitting to reflection-related words, which is mitigated by external CoT guidance. Notably, our experiments indicate that for RLLMs, one-shot CoT consistently yields superior performance compared to Few-shot CoT approaches. Our findings provide important insights for optimizing RLLMs’ performance through appropriate prompting strategies.

## 1 Introduction

Recent advances in Large Language Models (LLMs) have introduced Reasoning Large Language Models (RLLMs) such as OpenAI o1 (2024), DeepSeek-R1 (2025), and Qwen QwQ

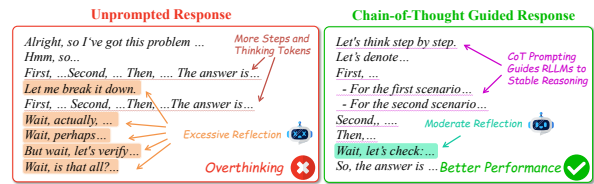


Figure 1: CoT prompting continues to play an important role in reasoning LLMs: (1) improving reasoning performance, (2) controlling the number of thinking tokens, (3) regulating the number of reasoning steps, and (4) mitigating overthinking.

(2025). These models generate innate Chain-of-Thought (CoT) before answering at inference time, demonstrating the effectiveness of test-time scaling (Muennighoff et al., 2025). Innate CoT is characterized by reflection and self-correction, which can significantly enhance a model’s ability to solve complex reasoning tasks (Kumar et al., 2024).

Over the past three years, as one of the most effective reasoning methods, CoT prompting (Wei et al., 2022) has been widely applied to LLMs and has helped models externally establish chain-like thinking for reasoning problems. This approach has significantly improved performance across various reasoning tasks (Zhang et al., 2022; Wang et al., 2022; Lyu et al., 2023; Ge et al., 2024). However, researchers from DeepSeek Guo et al. (2025) have expressed concern that Few-shot CoT might actually impair the performance of reasoning LLMs, implying that Few-shot CoT may not benefit RLLMs as it does traditional LLMs. This raises a concern: “Is CoT prompting, a popular In-Context Learning (ICL) method for chat LLMs, necessary to enhance the reasoning capability of RLLMs?”

In this paper, we present the first comprehensive analysis examining the impact of Zero-shot CoT (Kojima et al., 2022) & Few-shot CoT (Wei et al., 2022) on RLLMs for mathematical problems. Our experiments examine models with parameter sizes

ranging from 1.5B to 32B, revealing that Zero-shot CoT & Few-shot CoT significantly enhance the performance of reasoning LLMs on mathematical tasks in most cases. Notably, for large-capacity models: the improvement on simple datasets is minimal; however, the improvement on complex datasets is substantial; whereas for small-capacity models, the situation is reversed. Additionally, we find that Zero-shot CoT & Few-shot CoT play a significant role in regulating the numbers of thinking tokens and reasoning steps. To explore why direct prompting generates so many thinking tokens, we have statistically analyzed the reflection frequency of RLLM outputs. Surprisingly, on complex datasets, the average frequency of reflection per question is as high as over 800 for the 1.5B capacity model and over 400 for the 32B capacity model. It is worth noting that CoT prompting can reduce the average number of reflections by about 90% in some cases. To explore the reason for RLLM’s excessive reliance on reflection, we conducted deep analysis of the attention logits (before softmax) and attention scores (after softmax) of R1-8B and found its particular focus on words such as “Wait,” “Double-Check,” and “Ensure.” In contrast, its base model, LLAMA3.1-8B-INSTRUCT, did not exhibit such special attention. Thus, we believe that the reason for this overthinking is the RLLM’s overfitting to reflection and self-correction mechanisms. Through external prompt guidance, such as Zero-shot CoT & Few-shot CoT, the overfitting phenomenon can be alleviated. After observing the significant impact of Zero-shot CoT & Few-shot CoT on the performance of RLLMs, we conducted experiments to test the effect of the number of shots on performance. Unlike the common trend in LLMs where more shots generally lead to better performance, we found that one-shot CoT achieved the best performance across all datasets and models.

Our main contributions are as follows:

- We are the first to conduct a comprehensive analysis examining the impact of Zero-shot CoT & Few-shot CoT on RLLMs for mathematical problems.
- In terms of breadth, our experiments have revealed that Zero-shot CoT & Few-shot CoT plays a crucial role in controlling the distribution of the numbers of thinking tokens and reasoning steps of RLLM, as well as in suppressing overthinking.
- In terms of depth, we conducted a visualization

analysis of the attention logits of RLLMs and discovered the overfitting of RLLM to reflection words. Additionally, our findings indicate that one-shot CoT achieved the best performance for RLLMs.

## 2 Does CoT Prompting Still Matter for Reasoning LLMs?

In this section, we first highlight our experimental findings, then introduce our experimental setup, followed by details of each experiment and data analysis.

We begin by highlighting some of the most exciting results from our analysis here:

- In most cases, CoT prompting plays important roles in improving the performance of reasoning LLMs. The magnitude of improvement is influenced by the model’s parameter size and the difficulty of the dataset.
- RLLMs suffer from serious overthinking, the average frequency of reflection per question is as high as over 800 for the 1.5B capacity model and over 400 for the 32B capacity model on complex datasets. CoT prompting can effectively alleviate this issue.
- Setting the number of shots to 1 provides the maximum performance of RLLMs.

### 2.1 Preliminary

We employ three external CoT prompting methods in our experiments and briefly introduce these methods here: (1) **Direct**: only provides the problem description directly (without the <think> token). (2) **Zero-shot CoT** (Kojima et al., 2022): involves appending a thought inducing phrase “*Let’s think step by step.*” (3) **Few-shot CoT** (Wei et al., 2022): provides the LLM with a few exemplars, including task descriptions and expected outputs, to guide its reasoning. See Appendix B.1 for details.

### 2.2 Setup

**Models** Our experimental subjects are open-source reasoning LLMs: DeepSeek’s DEEPSEEK-R1-DISTILL-QWEN-1.5, DEEPSEEK-R1-DISTILL-QWEN-7B, DEEPSEEK-R1-DISTILL-LLAMA-8B, DEEPSEEK-R1-DISTILL-QWEN-14B, DEEPSEEK-R1-DISTILL-QWEN-32B, abbreviated as R1-1.5B, R1-7B, R1-8B, R1-14B, R1-32B respectively (Guo et al., 2025). Additionally, we included open-source models from the community: OPENO1-LLAMA-8B-V0.1 (Open

MODEL	PROMPT	GSM8K	ASDiv	SAT_MATH	MATH	AIME24	AMC23
DEEPSEEK-R1 SERIES							
R1-1.5B	Direct	5.7 <sub>(-)</sub>	11.7 <sub>(-)</sub>	46.9 <sub>(-)</sub>	14.4 <sub>(-)</sub>	3.3 <sub>(-)</sub>	10.0 <sub>(-)</sub>
	Few-shot CoT	31.3 <sub>(↑449.1)</sub>	<b>50.9</b> <sub>(↑335.0)</sub>	<b>93.8</b> <sub>(↑100.0)</sub>	<b>55.4</b> <sub>(↑284.7)</sub>	<b>6.7</b> <sub>(↑103.0)</sub>	<b>40.0</b> <sub>(↑300.0)</sub>
	Zero-shot CoT	<b>32.8</b> <sub>(↑475.4)</sub>	43.8 <sub>(↑274.4)</sub>	71.9 <sub>(↑53.3)</sub>	37.7 <sub>(↑161.8)</sub>	3.3 <sub>(0.0)</sub>	30.0 <sub>(↑200.0)</sub>
R1-7B	Direct	35.7 <sub>(-)</sub>	60.2 <sub>(-)</sub>	81.2 <sub>(-)</sub>	29.4 <sub>(-)</sub>	6.7 <sub>(-)</sub>	17.5 <sub>(-)</sub>
	Few-shot CoT	<b>81.1</b> <sub>(↑127.2)</sub>	<b>88.8</b> <sub>(↑47.5)</sub>	<b>96.9</b> <sub>(↑19.3)</sub>	<b>67.2</b> <sub>(↑128.6)</sub>	<b>20.0</b> <sub>(↑198.5)</sub>	<b>57.5</b> <sub>(↑228.6)</sub>
	Zero-shot CoT	69.6 <sub>(↑95.0)</sub>	70.8 <sub>(↑17.6)</sub>	78.1 <sub>(↓3.8)</sub>	65.8 <sub>(↑123.8)</sub>	6.7 <sub>(0.0)</sub>	42.5 <sub>(↑142.9)</sub>
R1-8B	Direct	78.2 <sub>(-)</sub>	84.9 <sub>(-)</sub>	65.6 <sub>(-)</sub>	64.3 <sub>(-)</sub>	<b>36.7</b> <sub>(-)</sub>	<b>62.5</b> <sub>(-)</sub>
	Few-shot CoT	69.1 <sub>(↓11.6)</sub>	81.7 <sub>(↓3.8)</sub>	<b>87.5</b> <sub>(↑33.4)</sub>	66.7 <sub>(↑3.7)</sub>	23.3 <sub>(↓36.5)</sub>	50.0 <sub>(↓20.0)</sub>
	Zero-shot CoT	<b>79.4</b> <sub>(↑1.5)</sub>	<b>85.1</b> <sub>(↑0.2)</sub>	84.4 <sub>(↑28.7)</sub>	<b>68.4</b> <sub>(↑6.4)</sub>	3.3 <sub>(↓91.0)</sub>	52.5 <sub>(↓16.0)</sub>
R1-14B	Direct	82.9 <sub>(-)</sub>	76.3 <sub>(-)</sub>	71.9 <sub>(-)</sub>	35.3 <sub>(-)</sub>	6.7 <sub>(-)</sub>	15.0 <sub>(-)</sub>
	Few-shot CoT	<b>89.8</b> <sub>(↑8.3)</sub>	<b>93.7</b> <sub>(↑22.8)</sub>	87.5 <sub>(↑21.7)</sub>	<b>72.1</b> <sub>(↑104.3)</sub>	<b>33.0</b> <sub>(↑392.5)</sub>	<b>70.0</b> <sub>(↑366.7)</sub>
	Zero-shot CoT	82.2 <sub>(↓0.8)</sub>	82.3 <sub>(↑7.9)</sub>	<b>90.6</b> <sub>(↑25.9)</sub>	61.5 <sub>(↑74.2)</sub>	13.3 <sub>(↑98.5)</sub>	37.5 <sub>(↑150.0)</sub>
R1-32B	Direct	82.7 <sub>(-)</sub>	85.7 <sub>(-)</sub>	81.2 <sub>(-)</sub>	42.3 <sub>(-)</sub>	10.0 <sub>(-)</sub>	20.0 <sub>(-)</sub>
	Few-shot CoT	83.2 <sub>(↑0.6)</sub>	<b>92.5</b> <sub>(↑7.9)</sub>	<b>100.0</b> <sub>(↑23.2)</sub>	<b>79.0</b> <sub>(↑86.8)</sub>	<b>43.3</b> <sub>(↑333.0)</sub>	<b>57.5</b> <sub>(↑187.5)</sub>
	Zero-shot CoT	<b>92.0</b> <sub>(↑11.3)</sub>	90.1 <sub>(↑5.1)</sub>	81.2 <sub>(0.0)</sub>	75.6 <sub>(↑78.7)</sub>	13.3 <sub>(↑33.0)</sub>	55.0 <sub>(↑175.0)</sub>
COMMUNITY MODELS							
OPENDO-8B/MARCO-7B	Direct	52.4 <sub>(-)</sub>	55.7 <sub>(-)</sub>	56.2 <sub>(-)</sub>	47.8 <sub>(-)</sub>	0.0 <sub>(-)</sub>	32.5 <sub>(-)</sub>
	Few-shot CoT	35.8 <sub>(↓31.7)</sub>	55.7 <sub>(0.0)</sub>	<b>78.1</b> <sub>(↑39.0)</sub>	<b>57.1</b> <sub>(↑19.5)</sub>	3.3 <sub>(↑∞)</sub>	<b>40.0</b> <sub>(↑23.1)</sub>
	Zero-shot CoT	<b>55.6</b> <sub>(↑6.1)</sub>	<b>59.2</b> <sub>(↑6.3)</sub>	56.2 <sub>(0.0)</sub>	38.2 <sub>(↓20.1)</sub>	<b>6.7</b> <sub>(↑∞)</sub>	20.0 <sub>(↓38.5)</sub>
OPENDO-8B/MARCO-7B	Direct	<b>74.9</b> <sub>(-)</sub>	77.0 <sub>(-)</sub>	75.0 <sub>(-)</sub>	36.1 <sub>(-)</sub>	<b>3.3</b> <sub>(-)</sub>	<b>32.5</b> <sub>(-)</sub>
	Few-shot CoT	71.2 <sub>(↓4.9)</sub>	<b>79.2</b> <sub>(↑2.9)</sub>	<b>84.4</b> <sub>(↑12.5)</sub>	<b>45.4</b> <sub>(↑25.8)</sub>	<b>3.3</b> <sub>(0.0)</sub>	17.5 <sub>(↓46.2)</sub>
	Zero-shot CoT	<b>74.9</b> <sub>(0.0)</sub>	<b>79.2</b> <sub>(↑2.9)</sub>	78.1 <sub>(↑4.1)</sub>	36.1 <sub>(0.0)</sub>	<b>3.3</b> <sub>(0.0)</sub>	22.5 <sub>(↓30.8)</sub>

Table 1: Accuracy (%) of various RLLMs across multiple datasets under different prompting settings: Direct (baseline), Few-shot CoT, and Zero-shot CoT. For non-baseline methods, performance changes compared to Direct prompting are shown below as percentages (%). For Few-shot CoT, the default number of shots is 5. Bold numbers represent the highest accuracy achieved for each model-dataset combination across the three prompting methods.

Source O1, 2024), MARCO-O1 (Zhao et al., 2024), abbreviated as OPENDO-8B, MARCO-7B. For detailed decoding configurations, see Appendix B.2.

**Datasets** We conduct our experiments on six mainstream English mathematical benchmarks, which cover difficulty levels ranging from elementary school to competition level: GSM8K (Cobbe et al., 2021), ASDiv (Miao et al., 2021), SAT\_MATH (Zhong et al., 2023), MATH (Hendrycks et al., 2021), AIME2024 (AI-MO, 2024a), and AMC2023 (AI-MO, 2024b).

**Metrics** We employed four metrics to analyze the experimental results: (1) Accuracy: The ratio of correct samples to the total number of samples; (2) Number of thinking tokens: The outputs of RLLMs comprise thinking and result parts. 'Thinking tokens' refers to the token count within the thinking component; (3) Number of reasoning steps: The

number of steps contained in the thinking parts. For example, "Firstly,...; Secondly, ...; Finally, ..." contains three steps; and (4) Number of reflections: The number of reflections per instance. For instance, "Wait, ..." constitutes one reflection. See Appendix B.3 for details.

### 2.3 The Impact of CoT prompting on Accuracy of Reasoning LLMs

As shown in Table 1, in 72% of cases, Zero-shot CoT and Few-shot CoT prompting continue to have a general impact on improving the accuracy of reasoning LLMs. Notably, CoT prompting demonstrates significant performance enhancements in specific model and dataset combinations. For instance, Zero-shot CoT improved R1-1.5B's performance on the GSM8K dataset by 475.4%, while Few-shot CoT enhanced R1-14B's performance on the AIME24 dataset by 392.5%. For large-capacity models, the improvement on sim-

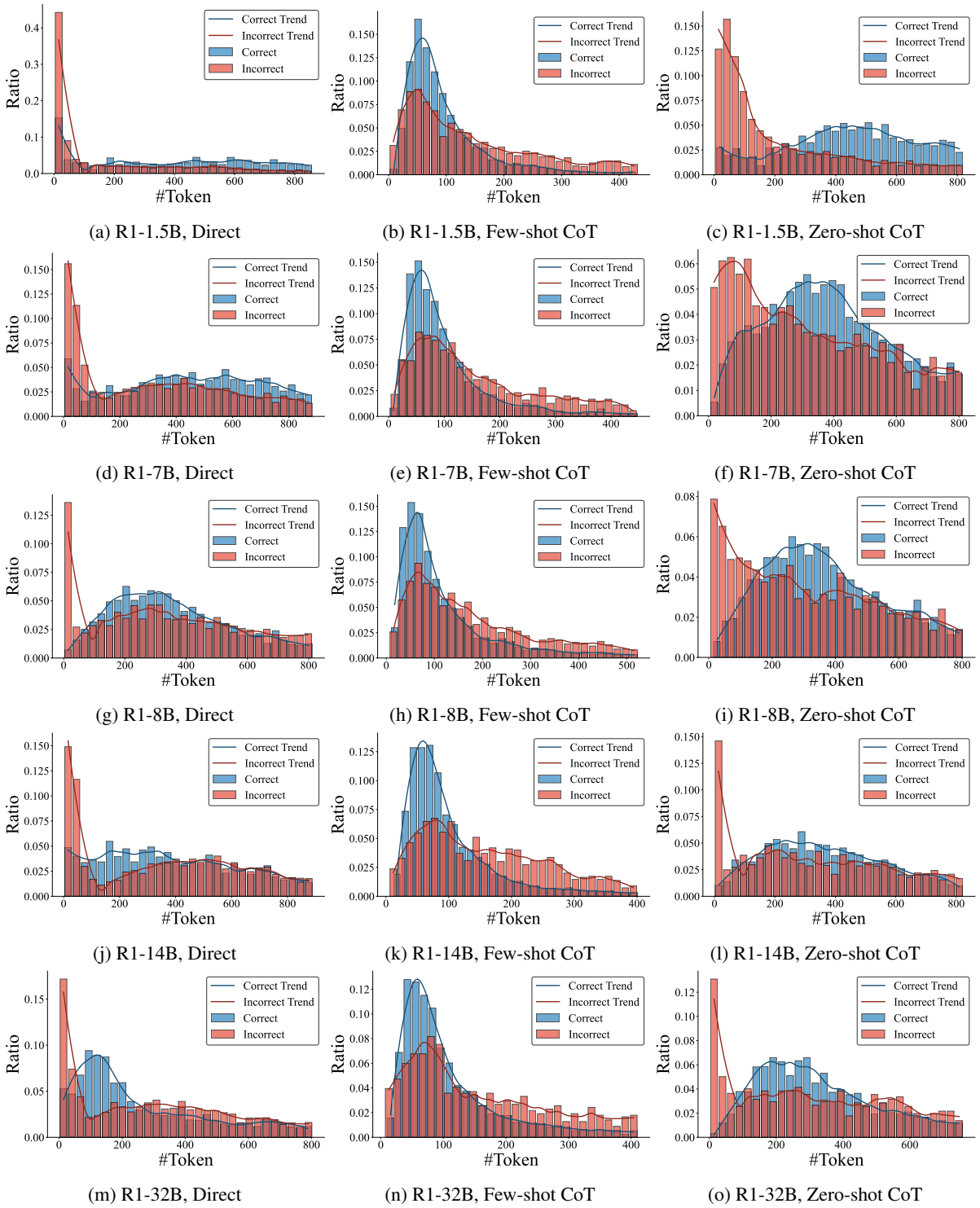


Figure 2: Distributions of thinking tokens across various RLLMs under three prompting methods evaluated on the MATH benchmark. The horizontal axis indicates the number of thinking tokens in the thinking parts (#Token), and the vertical axis represents the corresponding ratio. Histograms labeled “Correct” and “Incorrect” depict the distribution of token counts for correctly and incorrectly solved problems, while the trend lines (“Correct Trend” and “Incorrect Trend”) represent smoothed regression fits of these distributions.

211 ple datasets is minimal; however, the improve- 216  
 212 ment on complex datasets is substantial. Con- 217  
 213 versely, for small-capacity models, this situa- 218  
 214 tion is reversed. Taking R1-32B as an example of a 219  
 215 large-capacity model, it shows improvements of up

to 23.1% and as low as 0.6% on simple datasets 216  
 such as GSM8K, ASDiv, and SAT\_MATH. How- 217  
 ever, on complex datasets like MATH, AIME24, 218  
 and AMC23, R1-32B demonstrated improvements 219  
 ranging from 33% to 333%. For small-capacity 220

models, exemplified by R1-1.5B, improvements on the same simple datasets ranged from 53.3% to 475.44%, while on complex datasets, improvements ranged from no enhancement to up to 300%.

## 2.4 The Distribution of Thinking Tokens

As illustrated in Figure 2, the distributions of thinking tokens across DeepSeek series models are presented. From the analysis of prompting differences, under Direct prompting, the token distribution is highly dispersed, with numerous instances where the number of thinking tokens is less than 30. Few-shot CoT effectively regulates token distribution, with a substantial concentration of correct samples at approximately 100 tokens. This phenomenon can be attributed to LLMs’ tendency to emulate the examples provided in Few-shot CoT prompts. The token distribution for Zero-shot CoT can be interpreted as an intermediate state between Direct and Few-shot CoT: samples with extremely few thinking tokens persist, while simultaneously exhibiting clusters of correct samples concentrated within specific ranges of thinking token counts. This indicates that CoT prompting not only influences accuracy but also affects the distribution of the number of thinking tokens. Additionally, from the perspective of model capacity, under both Direct and Zero-shot CoT conditions, the primary distribution of thinking token quantities decreases as model capacity increases.

Paradoxically, we observe that beyond a certain threshold in token distribution, accuracy actually decreases as the number of output tokens increases. This phenomenon appears to diverge from previous research findings (Muennighoff et al., 2025; Jin et al., 2024), which led us to conduct more in-depth experiments in the following sections.

## 2.5 The Relationship Between Number of Reasoning Steps and Accuracy

As shown in Figure 3, across two complex datasets under two prompting settings, there exists a generally proportional relationship between the average numbers of thinking steps and accuracy. Notably, the slope of the trend line for Few-shot CoT exceeds that of Direct. Additionally, the distribution of step counts in Few-shot CoT tends to be smaller.

However, this does not resolve our confusion from the previous section, which prompted us to consider: *why does accuracy increase with additional steps while decreasing with additional tokens?* Upon analyzing the outputs of reasoning

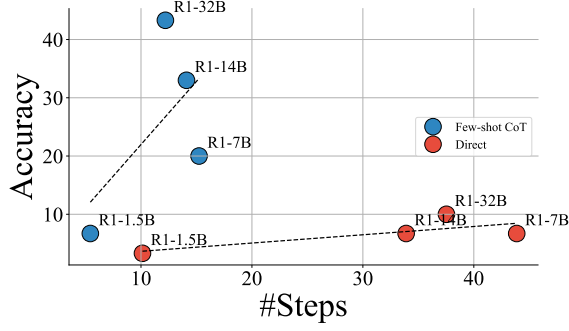
LLMs (see Appendix C for detail), we discovered the following phenomenon: reasoning LLMs engage in substantial reflection within individual reasoning steps to ensure answer correctness. Due to this reflection behavior, some responses contain few reasoning steps yet comprise numerous thinking tokens. This observation reminds us that the numbers of reasoning steps and thinking tokens are not proportionally related.

## 2.6 Excessive Reflection: The Unnecessary Exhaustion of Thinking Tokens

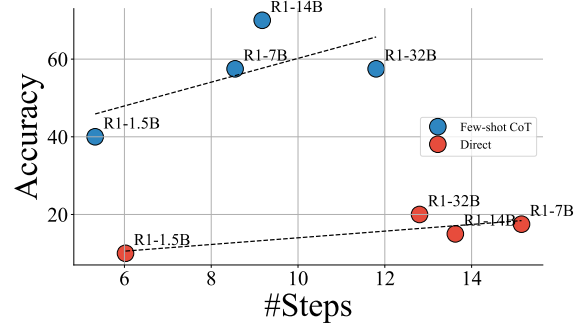
As shown in Table 2, responses from reasoning LLMs contain numerous reflection statements, indicating excessive self-correction and reflection. For example, R1-1.5B generates an average of 838.2 reflections per instance on the AIME24 dataset; while even the large-capacity model R1-32B averages 414.2 reflections per instance under the same conditions. Although AIME24 is a more challenging benchmark, generating hundreds of reflections per instance is clearly unreasonable. Furthermore, we observed a positive relationship between the average number of reflections per instance and dataset difficulty, meaning that as problem complexity increases, reasoning LLMs tend to produce even more frequent reflections. This aligns with our intuition: the more challenging the problem, the higher the model’s perplexity, leading to increased self-correction and reflection.

For instance, after implementing Few-shot CoT, R1-32B’s average reflections number per instance decreased from 414.2 to 2.56, while accuracy increased from 10% to 43.3%. It demonstrates that the majority of reflections per instance are redundant and produce unnecessary thinking tokens.

Furthermore, Zero-shot CoT demonstrates a stronger inhibitory effect on excessive self-correction and reflection compared to Few-shot CoT when applied to complex datasets. For example, on the AIME24 dataset, R1-14B averages 63.77 reflections per instance with Few-shot CoT, whereas with Zero-shot CoT, this average decreases to 8.80. Similarly, on the AMC23 dataset, R1-32B averages 101.85 reflections per instance with Few-shot CoT, while with Zero-shot CoT, this average is reduced to 4.17. These findings indicate that employing Zero shot CoT is a simpler and token efficient method for suppressing overthinking.



(a) AIME24



(b) AMC23

Figure 3: Relationship between accuracy and the average number of reasoning steps for different RLLMs evaluated on AIME24 and AMC23. The horizontal axis represents the average number of reasoning steps (#Steps), and the vertical axis represents accuracy. Dotted lines indicate regression fits illustrating the general correlation trends between average number of reasoning steps and accuracy.

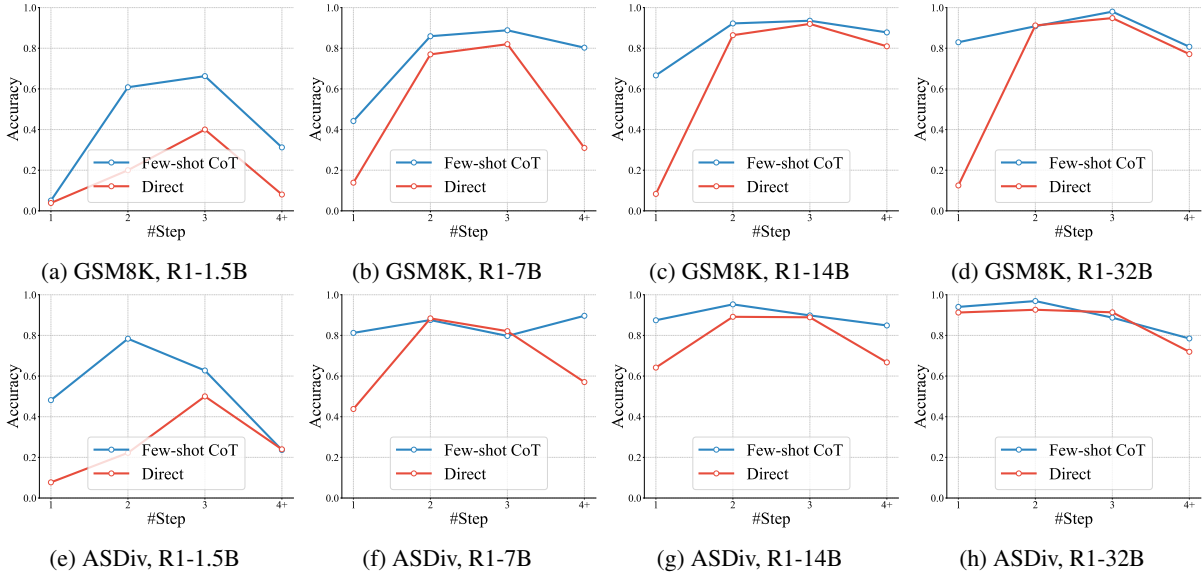


Figure 4: Relationship between the number of reasoning steps (#Step) and accuracy of RLLMs on the GSM8K and ASDiv datasets. The accuracy is averaged across individual reasoning steps provided by the RLLMs. Results show that accuracy initially increases with the number of steps but declines after reaching an optimal point (around 2-3 steps).

### 3 Deeper Exploration

#### 3.1 Overthinking is Commonplace

Our previous analysis identified excessive reflections as a significant phenomenon in reasoning LLMs tackling complex mathematical problems. This behavior manifests as numerous reflections that increase token counts without proportionally improving accuracy. We now examine whether this behavior persists in simpler mathematical problems.

As shown in Figure 4, we analyzed the relationship between accuracy and reasoning steps across different model capacities on simpler datasets

(GSM8K and ASDiv). The results reveal a consistent pattern: accuracy initially increases with additional reasoning steps but begins to decline after reaching an optimal point (typically 2-3 steps). This inverted U-shaped relationship is particularly pronounced in smaller models. For example, in GSM8K with the R1-1.5B model (Figure 4a), accuracy peaks at 3 steps under Few-shot CoT prompting before dropping significantly at 4 steps. Similarly, in ASDiv, optimal performance occurs at 2 steps before declining. Since these datasets typically require no more than 3 steps to solve, additional steps represent redundant reasoning

MODEL	AIME24			AMC23			MATH		
	DIRECT	5-SHOT	0-CoT†	DIRECT	5-SHOT	0-CoT	DIRECT	5-SHOT	0-CoT†
R1-1.5B	838.2	90.3(↓89.2)	18.1(↓97.8)	497.5	170.7(↓65.7)	15.6(↓96.9)	23.5	5.0(↓78.8)	13.9(↓40.9)
R1-7B	435.1	167.8(↓61.4)	6.2(↓98.6)	406.4	123.6(↓69.6)	5.1(↓98.7)	14.2	2.0(↓86.0)	4.7(↓66.9)
R1-8B	59.9	49.2(↓18.0)	10.2(↓83.0)	41.4	60.6(↑46.4)	4.8(↓88.4)	2.0	1.5(↓25.6)	3.4(↑74.9)
R1-14B	529.0	63.8(↓87.9)	8.8(↓98.3)	307.9	52.4(↓83.0)	6.6(↓97.9)	10.5	1.6(↓84.6)	4.4(↓58.5)
R1-32B	414.2	2.6(↓99.4)	7.8(↓98.1)	268.0	101.9(↓62.0)	4.2(↓98.4)	6.9	1.1(↓83.5)	3.0(↓57.0)
QWEN2.5-MATH	0.0	0.0(0.0)	0.0(0.0)	0.0	0.0(0.0)	0.0(0.0)	0.1	0.0(↓33.3)	0.0(↓83.3)
LLAMA3.1-8B	0.0	0.0(0.0)	0.0(0.0)	0.0	0.0(0.0)	0.0(0.0)	0.2	0.0(↓100.0)	0.5(↑113.6)

Table 2: Average number of reflections per instance across different LLMs under three prompting settings on three mathematical datasets. Values shown in gray indicate results under the Direct baseline. For non-baseline methods, relative performance changes are shown below as percentages (%). Models QWEN2.5-MATH-7B and LLAMA3.1-8B-INSTRUCT, abbreviated as QWEN2.5-MATH and LLAMA3.1-8B respectively, are included as comparative baselines since they serve as the base models from which R1-7B and R1-8B are fine-tuned. †: 0-CoT refers to Zero-shot CoT.

#SHOTS	R1-1.5B	R1-7B	R1-14B	R1-32B
0	3.3(—)	6.7(—)	6.7(—)	10.0(—)
1	<b>13.3</b> (↑ 303.0)	<b>23.3</b> (↑ 247.8)	<b>36.7</b> (↑ 447.8)	<b>56.7</b> (↑ 467.0)
2	3.3(0.0)	10.0(↑ 49.3)	26.7(↑ 298.5)	40.0(↑ 300.0)
3	10.0(↑ 203.0)	20.0(↑ 198.5)	23.3(↑ 247.8)	30.0(↑ 200.0)
4	6.7(↑ 103.0)	<b>23.3</b> (↑ 247.8)	33.3(↑ 397.0)	43.3(↑ 333.0)
5	6.7(↑ 103.0)	20.0(↑ 198.5)	33.0(↑ 392.5)	43.3(↑ 333.0)

Table 3: Accuracy (%) of RLLMs on the AIME24 dataset under different Few-shot CoT settings. “#Shots” indicates the number of Question-Answer pairs as examples provided to the model. The baseline (Direct, without any Question-Answer pair as example) is shaded in grey, with percentages below showing relative performance changes (%) compared to this baseline. Bold numbers represent the highest accuracy achieved for each model.

rather than productive problem-solving. This phenomenon appears universally across model sizes, though its severity varies. Smaller models (R1-1.5B and R1-7B) experience more dramatic performance degradation with excessive steps, while larger models (R1-14B and R1-32B) maintain relatively stable performance even with additional steps. These findings confirm that excessive reflection remains prevalent even in simpler datasets, and that adding steps beyond necessity does not improve and often harms accuracy.

### 3.2 Impact of the Number of Shots on RLLM Performance

In the preceding sections, our experiments have demonstrated that CoT prompting significantly enhances the performance of reasoning LLMs across most scenarios. However, in our default experimental configuration, we utilized a five-shot setting for Few-shot CoT prompting. This raises a ques-

tion: What is the optimal number of exemplars for maximizing RLLM performance, and how does performance vary as the number of shots changes?

Table 3 presents the accuracy of various DeepSeek models on the challenging AIME24 dataset under different Few-shot CoT settings (0-5 shots). The results reveal a clear pattern: providing exactly one Question-Answer pair (one-shot) yields optimal or near-optimal performance for most model sizes. For R1-1.5B, R1-14B, and R1-32B, one-shot CoT prompting produces the highest accuracy, with improvements ranging from 303.0% for R1-1.5B to 467.0% for R1-32B compared to the Direct baseline. This finding suggests that minimal exemplification—just a single example—provides sufficient structural guidance for most RLLMs to navigate complex reasoning tasks. Additional examples beyond this point rarely improve performance and often lead to degradation, particularly in the 2-3 shot range. This pattern indicates that

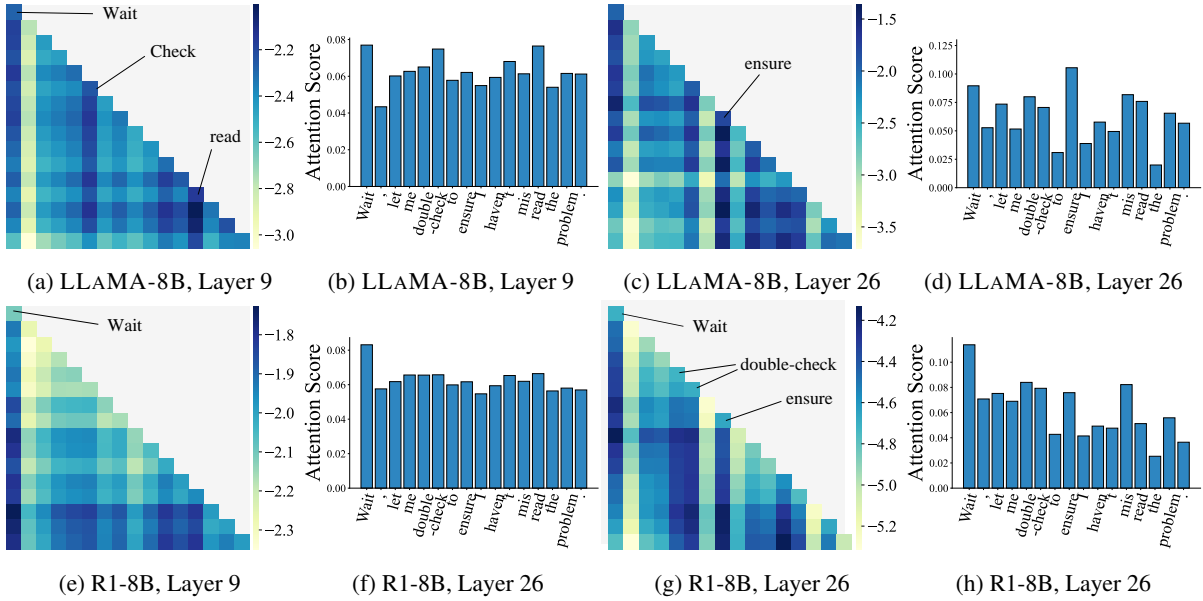


Figure 5: Visualization of attention distribution in LLAMA3.1-8B-INSTRUCT and R1-8B. The heatmaps (left side) show attention logits (before softmax), averaged over all heads per layer, and the corresponding bar graphs (right side) illustrate the softmax-normalized attention scores for the input sequence “Wait, let me double-check to ensure I haven’t misread the problem.” Subfigures (a)-(d) represent the LLAMA3.1-8B-INSTRUCT at layers 9 and 26, while subfigures (e)-(h) depict the R1-8B at the same layers. Here, the attention scores, denoted by  $\alpha$ , are computed as  $\alpha = \mathbb{E}_h [\sigma(\mathbf{A})]$ .

RLLMs may struggle with interference from multiple examples, with a single clear example providing the optimal balance between guidance and flexibility.

### 3.3 Attention-Based Insights into Over-Reflection

To investigate the mechanistic origins of excessive reflection, we analyzed attention patterns in R1-8B and its base model LLAMA3.1-8B-INSTRUCT. Figure 5 visualizes their attention distribution mechanisms when processing a typical reflection phrase.

The visualizations reveal that R1-8B allocates significantly higher attention to reflection tokens compared to LLAMA3.1-8B-INSTRUCT. At both middle (layer 9) and deep (layer 26) layers, R1-8B exhibits intensified attention logits for tokens such as “Wait” and “double-check” (Figures 5e and 5g). This pattern is further confirmed in the normalized attention scores, where R1-8B allocates substantially higher attention to “Wait” (Figures 5f and 5h) than LLAMA3.1-8B-INSTRUCT (Figures 5b and 5d). These observations suggest that reasoning LLMs have demonstrated an oversensitivity to linguistic markers of verification and reassessment during their training process. This hyperattention to reflection cues likely contributes to the excessive

reflection behavior observed in our experiments. The phenomenon appears to stem from an unintended consequence of RLLM training, wherein models overfit to reflection-related keywords.

## 4 Conclusion

Our study provides the first comprehensive analysis of CoT prompting for RLLMs, addressing concerns about its potential negative impact. Our experiments across models from 1.5B to 32B parameters on various mathematical tasks demonstrate that both Zero-shot CoT and Few-shot CoT significantly enhance RLLM performance in most scenarios. Large-capacity models showed minimal improvement on simple tasks but substantial gains on complex problems, while smaller models exhibited the opposite pattern. Notably, one-shot prompting consistently outperforms multi-shot approaches. Additionally, CoT effectively regulates thinking token distribution and reasoning steps, reducing excessive reflection. Attention analysis revealed the mechanism behind this phenomenon: RLLMs overfit to reflection-related linguistic tokens, which CoT helps mitigate. Our findings provide crucial insights for optimizing RLLM performance through appropriate prompting strategies, confirming that external CoT remains vital for enhancing mathematical reasoning in RLLMs.

## 438 Limitations

439 While our study provides comprehensive analysis  
440 of CoT prompting effects on RLLMs, several limi-  
441 tations should be acknowledged. First, our experi-  
442 ments focus primarily on mathematical reasoning  
443 tasks, and the generalizability to other reasoning  
444 domains (e.g., commonsense reasoning, logical rea-  
445 soning) remains to be validated. Second, we eval-  
446 uate only open-source RLLMs due to API limita-  
447 tions, and our findings may not directly transfer to  
448 proprietary models like OpenAI o1. Third, our at-  
449 tention analysis is conducted on specific layers and  
450 may not capture the full complexity of attention  
451 patterns across all architectures. Finally, while we  
452 demonstrate that one-shot CoT outperforms multi-  
453 shot approaches, the optimal prompting strategy  
454 may vary across different task types and model  
455 configurations.

## 456 Ethics Statement

457 In conducting our research, we place paramount  
458 importance on ethical standards to ensure integrity  
459 and contribute positively to the scientific commu-  
460 nity. We exclusively utilize open-source datasets,  
461 ensuring that our work is built upon accessible  
462 and transparent resources. Our methods employ  
463 models that are either open-source or have gained  
464 wide recognition for their reliability and ethical use  
465 within the academic community. Furthermore, we  
466 have meticulously designed our methodology to  
467 prevent the generation of harmful or misleading  
468 information, thereby safeguarding the integrity of  
469 our findings.

## 470 References

471 AI-MO. 2024a. [Ai-mo/aimo-validation-aime](#).

472 AI-MO. 2024b. [Ai-mo/aimo-validation-amc](#).

473 Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian,  
474 Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias  
475 Plappert, Jerry Tworek, Jacob Hilton, Reiichiro  
476 Nakano, Christopher Hesse, and John Schulman.  
477 2021. Training verifiers to solve math word prob-  
478 lems. *arXiv preprint arXiv:2110.14168*.

479 Yuyao Ge, Shenghua Liu, Baolong Bi, Yiwei Wang,  
480 Lingrui Mei, Wenjie Feng, Lizhe Chen, and Xueqi  
481 Cheng. 2024. Can graph descriptive order affect  
482 solving graph problems with llms? *arXiv preprint*  
483 *arXiv:2402.07140*.

484 Daya Guo, Dejian Yang, Haowei Zhang, Junxiao  
485 Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shi-  
486 rong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025.

487 Deepseek-r1: Incentivizing reasoning capability in  
488 llms via reinforcement learning. *arXiv preprint*  
489 *arXiv:2501.12948*.

490 Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul  
491 Arora, Steven Basart, Eric Tang, Dawn Song, and Ja-  
492 cob Steinhardt. 2021. Measuring mathematical prob-  
493 lem solving with the math dataset. *arXiv preprint*  
494 *arXiv:2103.03874*.

495 Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao,  
496 Wenye Hua, Yanda Meng, Yongfeng Zhang, and  
497 Mengnan Du. 2024. [The impact of reasoning step length on large language models](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1830–1842, Bangkok, Thailand. Association for Computational Linguistics.

502 Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yu-  
503 taka Matsuo, and Yusuke Iwasawa. 2022. Large lan-  
504 guage models are zero-shot reasoners. *Advances in*  
505 *neural information processing systems*, 35:22199–  
506 22213.

507 Aviral Kumar, Vincent Zhuang, Rishabh Agarwal,  
508 Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli,  
509 Shariq Iqbal, Colton Bishop, Rebecca Roelofs, and  
510 1 others. 2024. Training language models to self-  
511 correct via reinforcement learning. *arXiv preprint*  
512 *arXiv:2409.12917*.

513 Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang,  
514 Delip Rao, Eric Wong, Marianna Apidianaki, and  
515 Chris Callison-Burch. 2023. Faithful chain-of-  
516 thought reasoning. In *The 13th International Joint*  
517 *Conference on Natural Language Processing and the*  
518 *3rd Conference of the Asia-Pacific Chapter of the*  
519 *Association for Computational Linguistics (IJCNLP-*  
520 *AAACL 2023)*.

521 Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su.  
522 2021. A diverse corpus for evaluating and developing  
523 english math word problem solvers. *arXiv preprint*  
524 *arXiv:2106.15772*.

525 Niklas Muennighoff, Zitong Yang, Weijia Shi, Xi-  
526 ang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke  
527 Zettlemoyer, Percy Liang, Emmanuel Candès, and  
528 Tatsunori Hashimoto. 2025. s1: Simple test-time  
529 scaling. *arXiv preprint arXiv:2501.19393*.

530 Open Source O1. 2024. Open O1: A Model Match-  
531 ing Proprietary Power with Open-Source Inno-  
532 vation. [https://github.com/Open-Source-O1/](https://github.com/Open-Source-O1/Open-O1)  
533 [Open-O1](#).

534 OpenAI. 2024. Learning to reason with  
535 llms. [https://openai.com/index/](https://openai.com/index/learning-to-reason-with-llms)  
536 [learning-to-reason-with-llms](#). Accessed:  
537 January 6, 2026.

538 Qwen. 2025. [Qwq-32b: The power of scaling rl](#). 538

539 Qwen Team. 2024. [Qwq: Reflect deeply on the bound-aries of the unknown](#). 540

- 541 Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le,  
542 Ed Chi, Sharan Narang, Aakanksha Chowdhery, and  
543 Denny Zhou. 2022. Self-consistency improves chain  
544 of thought reasoning in language models. *arXiv*  
545 *preprint arXiv:2203.11171*.
- 546 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten  
547 Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou,  
548 and 1 others. 2022. Chain-of-thought prompting elic-  
549 its reasoning in large language models. *Advances*  
550 *in neural information processing systems*, 35:24824–  
551 24837.
- 552 Yuyang Wu, Yifei Wang, Tianqi Du, Stefanie Jegelka,  
553 and Yisen Wang. 2025. When more is less: Un-  
554 derstanding chain-of-thought length in llms. *arXiv*  
555 *preprint arXiv:2502.07266*.
- 556 Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex  
557 Smola. 2022. Automatic chain of thought prompt-  
558 ing in large language models. *arXiv preprint*  
559 *arXiv:2210.03493*.
- 560 Yu Zhao, Huifeng Yin, Bo Zeng, Hao Wang, Tianqi  
561 Shi, Chenyang Lyu, Longyue Wang, Weihua Luo,  
562 and Kaifu Zhang. 2024. [Marco-01: Towards open](#)  
563 [reasoning models for open-ended solutions](#). *Preprint*,  
564 [arXiv:2411.14405](#).
- 565 Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang,  
566 Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen,  
567 and Nan Duan. 2023. Agieval: A human-centric  
568 benchmark for evaluating foundation models. *arXiv*  
569 *preprint arXiv:2304.06364*.

## A Related Works

### A.1 Research and Analysis on CoT Prompting

Chain-of-Thought prompting, first introduced by Wei et al. (2022), showed that providing exemplars of intermediate reasoning steps can significantly boost LLMs’ performance on complex tasks. Soon after, Kojima et al. (2022) discovered that even without any demonstrations, simply appending a prompt like "Let’s think step by step" enables strong Zero-shot CoT reasoning. Subsequent efforts focused on automating and refining CoT prompts. Zhang et al. (2022) proposed Auto-CoT, which automatically generates diverse reasoning chains for Few-shot prompts. In parallel, Wang et al. (2022) introduced a self-consistency decoding strategy: by sampling multiple distinct reasoning paths and selecting the most consistent final answer, they achieved striking performance gains in CoT prompting. Beyond new prompting strategies, researchers also analyzed how CoT content affects outcomes. Jin et al. (2024) found that longer reasoning sequences, even containing minor mistakes, substantially enhance LLM reasoning accuracy, whereas overly concise chains degrade it. Building upon this work, Wu et al. (2025) demonstrate that a nuanced relationship exists between CoT length and performance, identifying an optimal length that balances decomposition benefits against error accumulation based on model capability and task complexity.

### A.2 Reasoning Large Language Models with Innate CoT

Despite the excellent performance of CoT prompting, inherent limitations in adaptability persist; consequently, reasoning LLMs, such as OpenAI’s o1 (2024) have been introduced to generate reasoning internally, offering enhanced efficiency and broader generality. Shortly after its introduction, the research community responded with projects: Open-O1 (2024). Subsequently, Alibaba launched both Marco-O1 (2024) and QWQ-preview (2024). The former integrates search algorithms and reflective prompting within a small-scale model to achieve step-by-step problem solving despite limited resources, whereas the latter illustrates that medium-scale open-source models, when combined with reinforcement learning, can approach the reasoning capabilities of larger proprietary models. DeepSeek-R1 (2025) represents the apex of this evolutionary trajectory by adopting an extreme

“large-scale + pure reinforcement learning” route to achieve reasoning performance on par with OpenAI’s O1 models. However, researchers from DeepSeek Guo et al. (2025) have expressed concern that Few-shot CoT might impair the performance of RLLMs, implying that Few-shot CoT may not benefit RLLMs as it does traditional LLMs. The concern sparked our curiosity. Although previous research has explored the impact of CoT prompting on LLMs, our work is the first detailed study on the influence of CoT prompting on reasoning LLMs.

## B Detail of Experiment

### B.1 Prompt

Table 4 presents detailed prompt templates for DeepSeek series models distilled from the Qwen family (e.g., R1-1.5B, R1-7B). When adapting to a LLaMA-based model (e.g., R1-8B, LLAMA3.1-8B-INSTRUCT), the template undergoes a replacement of “Question:” with “User:” and “Answer:” with “Assistant:”. For MARCO-7B, the template undergoes a replacement of “Question:” with “User:” and “Answer:” with “Content:”.

Below we provide examples of the CoT prompting templates used in our experiments. These examples demonstrate how we formatted different types of mathematical problems and their corresponding CoT solutions across various datasets.

For simpler datasets like GSM8K and ASDiv, we selected question-answer pairs that demonstrate basic arithmetic reasoning, as shown in the following example:

Example from GSM8K
Question: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?
Answer: There are 15 trees originally. Then there were 21 trees after some more were planted. So there must have been $21 - 15 = 6$ . The answer is 6.

For more complex datasets like AMC23, AIME24 and MATH, we utilized examples that demonstrate advanced mathematical reasoning with explicit step-by-step solutions:

### Example from MATH

Question: Kevin Kangaroo begins hopping on a number line at 0. He wants to get to 1, but he can hop only  $\frac{1}{3}$  of the distance. Each hop tires him out so that he continues to hop  $\frac{1}{3}$  of the remaining distance. How far has he hopped after five hops? Express your answer as a common fraction.

Answer: Let's think step by step Kevin hops  $\frac{1}{3}$  of the remaining distance with every hop. His first hop takes  $\frac{1}{3}$  closer. For his second hop, he has  $\frac{2}{3}$  left to travel, so he hops forward  $(\frac{2}{3})(\frac{1}{3})$ . For his third hop, he has  $(\frac{2}{3})^2$  left to travel, so he hops forward  $(\frac{2}{3})^2(\frac{1}{3})$ . In general, Kevin hops forward  $(\frac{2}{3})^{k-1}(\frac{1}{3})$  on his  $k$ th hop. We want to find how far he has hopped after five hops. This is a finite geometric series with first term  $\frac{1}{3}$ , common ratio  $\frac{2}{3}$ , and five terms. Thus, Kevin has hopped  $\frac{\frac{1}{3}(1-(\frac{2}{3})^5)}{1-\frac{2}{3}} = \frac{211}{243}$ .

The answer is  $\frac{211}{243}$ .

For multiple choice datasets like SAT\_MATH, we included multiple-choice format examples:

### Example from SAT\_MATH

Question: If  $\frac{x-1}{3} = k$  and  $k = 3$ , what is the value of  $x$ ?  
Answer Choices: (A) 2 (B) 4 (C) 9 (D) 10

Answer: If  $k = 3$ , then  $x - 1 = 3 \times 3$ , therefore,  $x - 1 = 9$  and  $x = 10$ . The answer is D.

## B.2 Implementation Details

We deploy the open-source LLMs for our experiments on a  $4 \times$  NVIDIA A800 server. The decoding temperature was set to zero (Greedy decoding). We set the number of maximum new tokens according to the level of datasets. Specifically, for simple datasets (GSM8K, ASDiv, SAT\_MATH), the number of maximum token per call is set to 2048; for complex datasets (AIME24, AMC23), it is set to 32768.

## B.3 Metrics

### Accuracy

$$\text{Accuracy} = \frac{\# \text{correct answers}}{\# \text{total questions}} \quad (1)$$

where # represents the number of instances.

**Number of Thinking Tokens** For OPENO1-8B, the thinking part is wrapped in '<Thought>' tags. For others, the content before the last final answer keywords is defined as thinking parts. The answer keywords contain: "the answer is", "The answer is", "Final Answer", "final answer is", "\*\*\*Final Answer", "\*\*\*Conclusion:\*\*", "\*\*\*Answer:\*\*".

**Number of Reasoning steps** We employ LLAMA3.1-8B-INSTRUCT to analyze the thinking part of RLLMs' responses, specifically to quantify the number of reasoning steps. The prompt templates are utilized as follows:

### Prompt Template for Answering The Number of Reasoning Steps

Analyze the following mathematical solution and count how many distinct thinking steps are used.

A step is defined as a logical unit where a specific calculation or deduction is made.

Equations that are directly derived from previous ones count as the same step if they're part of the same logical operation.

Here's a solution example:

To find the total meters James runs in a week, we need to break down the problem step by step. First, determine how many sprints James does each week. He runs 3 sprints 3 times a week, so the total number of sprints is 3 multiplied by 3, which equals 9 sprints.

Next, calculate the total distance by multiplying the number of sprints by the distance of each sprint. Each sprint is 60 meters, so 9 sprints multiplied by 60 meters per sprint equals 540 meters.

Therefore, James runs a total of 540 meters each week.

#### Solution:

To determine the total number of meters James runs in a week, follow these steps:

#### 1. Calculate the total number of sprints per week:

James runs 3 sprints each day and does this 3 times a week.

Total sprints per week = 3 sprints/day  $\times$  3 days = 9 sprints

#### 2. Calculate the total distance run:

Each sprint is 60 meters. Multiply the total number of sprints by the distance of each sprint.

Total distance = 9 sprints  $\times$  60 meters/sprint = 540 meters

#### Final Answer:

540 meters

For this example solution, the answer would be:

```
{{"num_steps": 2}}
```

Because there are 2 distinct thinking steps:

1. Calculating the total number of sprints per week.

2. Calculating the total distance run

Now analyze the following solution:

Solution to analyze:

<RLLMs' response content>

Provide only a JSON object with the following format:

```
{{"num_steps": <number of distinct thinking steps>}}
```

**Number of reflection** We quantify the frequency of reflective keywords within the thinking parts of RLLMs' responses.

The reflective keywords contain: "wait", "Let me think", "I need to reconsider", "Let me recalculate", "Let me double-check", "Let me verify", "Let me correct", "Let me revise", "Let me rethink",

Prompt Style	Prompt Template
Direct	Question: <Question> \n Answer:
Zero-shot CoT	Question: <Question> \n Answer: Let's think step by step.
Few-shot CoT	Question: <Example question> Answer: <Example Answer> \n ... (more Few-shot CoT examples) \n Question: <Question> \n Answer:

Table 4: Prompt styles and their corresponding templates for DeepSeek series models distilled from the Qwen family.

“Let me try again”, “Hold on”, “I see a problem”, “Let me backtrack”, “Let me restart”, “I need to rethink”, “Let me approach this differently”, “Let me redo”, “No, that’s not right”, “Let’s break down the reasoning”.

## C Case Study

To further illustrate our findings, we present a detailed case study comparing the reasoning process of the R1-32B model across different prompting methods. We selected a problem, as shown below, from the AIME24 dataset that was solved correctly under both Direct and Few-shot CoT settings.

### Pure Question from AIME24

Alice chooses a set  $A$  of positive integers. Then Bob lists all finite nonempty sets  $B$  of positive integers with the property that the maximum element of  $B$  belongs to  $A$ . Bob’s list has 2024 sets. Find the sum of the elements of  $A$ .

Under direct prompting, R1-32B generates 1088 thinking tokens across 15 reasoning steps, exhibiting extensive self-reflection patterns. The model frequently interrupts its own reasoning with phrases like “Wait” and “Let me check” (highlighted in red), demonstrating the excessive reflection tendency identified in our experimental analysis. Despite reaching the correct answer (55) midway through its reasoning, the model continues to question its conclusion, alternating between accepting and rejecting its own results before finally committing to the correct answer.

### Answer of R1-32B of Problem Described Directly

Okay, so... Hmm, so... Let’s break this down... Well, for... So, the number... Since... Therefore... Sum... We need... *Wait*, but 2024 is equal... So, if we can express... But 2024 is equal to... Hmm, but 24 is 16 + 8, which is... So, 2024 = ... *But wait*, that would mean... But in terms of binary... *Let me check*:...  $2^{10}$  is 1024,  $2^{11}$  is 2048. So... 24 is 16 + 8, so 2024 is... But in binary, 2024 is... *Let’s verify*: 1111101000 is:  $1 * 2^{10} = ...1 * 2^9 = ...1 * 2^8 = ...1 * 2^7 = ...1 * 2^6 = ...1 * 2^5 = ...0 * 2^4 = ...1 * 2^3 = ...0 * 2^2 = ...0 * 2^1 = ...0 * 2^0 = ...$  Adding these up... Yes, that’s correct. So, 2024 in binary is... which corresponds to... So, the exponents are... But in our earlier equation... So, each term... Therefore, the exponents... So,  $a - 1 = ...$  Therefore ... Therefore, the sum... *Let’s compute that*:  $4 + 6 = ...10 + 7 = ...17 + 8 = ...25 + 9 = ...34 + 10 = ...44 + 11 = ...$  So, the sum is 55. *Wait*, but the initial answer was 2024, which is wrong because... So, the initial thought was incorrect. Therefore, *the correct answer should be 2024*. *But let me double-check*. We have... Expressed in binary, 2024 is... which has 1s at positions... So, the exponents... so  $a = ...$  Compute step by step:  $4 + 6 = ...10 + 7 = ...17 + 8 = ...25 + 9 = ...34 + 10 = ...44 + 11 = 55$ . Yes, that’s correct. So, the sum is 55. Therefore, *the initial answer of 2024 was incorrect. The correct answer is 55*. *But wait, let me think again*. Is there another way... The problem says:... So, for each  $a$  in  $A$ ... Therefore, the total number... So, we need... So, 2024 is... Hence, *the answer is 55*. *But wait*, the initial answer was 2024, which is... So, the user... Therefore, *the correct answer is 55*. *But let me check once more*. Suppose  $A = ...$  Then, for each  $a$  in  $A$ ... So, for  $a=4: ... a=6: ... a=7: ... a=8: ... a=9: ... a=10: ... a=11: ...$  Now, let’s sum these up:  $8 + 32 = ...40 + 64 = ...104 + 128 = ...232 + 256 = ...488 + 512 = ...$  Yes, that’s correct. So, the total number of sets is indeed 2024, and the sum of  $A$  is 55. Therefore, *the answer is 55*. 55

In contrast, when prompted with Few-shot CoT, the same model produces only 420 thinking tokens across 12 reasoning steps—a 61.4% reduction in token consumption. The reasoning process displays markedly higher information density per step, with significantly fewer reflective interruptions. While

the model still engages in some verification (as seen in the “Wait, let’s compute” statement), the overall reasoning follows a more direct path toward the solution with minimal redundancy.

**Answer of R1-32B of Problem Described with Few-shot CoT Prompting**

Let’s think... Let’s denote... Each element... Therefore, the total... Given that...  $\sum_{a \in A} 2^{a-1} = 2024$ .  
 Let’s factor 2024:  $2024 \div 2 = \dots 1012 \div 2 = \dots 506 \div 2 = \dots 253 \div 11 = \dots$   
 So,  $2024 = 2^3 \times 11 \times 23$ .  
 We need... But 2024 in binary...  
*Wait, let’s compute 2024 in binary:*  
 $2024 \div 2 = \dots 1012 \div 2 = \dots 506 \div 2 = \dots 253 \div 2 = \dots 126 \div 2 = \dots 63 \div 2 = \dots 31 \div 2 = \dots 15 \div 2 = \dots 7 \div 2 = \dots 3 \div 2 = \dots 1 \div 2 = \dots$   
 So writing... 11111101000.  
 So  $2024 = 1024 + 512 + 256 + 128 + 64 + 32 + 8$ .  
*Wait, let’s check:*  
 $1024 + 512 = \dots 1536 + 256 = \dots 1792 + 128 = \dots 1920 + 64 = \dots 1984 + 32 = \dots 2016 + 8 = 2024$ .  
 Yes, correct.  
 So  $2024 = 2^{10} + 2^9 + 2^8 + 2^7 + 2^6 + 2^5 + 2^3$ .  
 But in our problem... So each term... Therefore, the exponents... Thus,  $a-1 = \dots$  Therefore,  $a = \dots$   
 So the set A is 4,6,7,8,9,10,11.  
 Therefore, the sum... Compute this:  
 $4 + 6 = \dots 10 + 7 = \dots 17 + 8 = \dots 25 + 9 = \dots 34 + 10 = \dots 44 + 11 = 55$ .  
 So the sum is 55.  
*The answer is 55*

This case study provides concrete evidence of how Few-shot CoT prompting effectively regulates reasoning behavior in RLLMs by reducing excessive reflections while maintaining or improving solution accuracy. The higher information density in the Few-shot example demonstrates that while the raw number of reasoning steps is only slightly reduced, the efficiency of those steps is substantially improved through the elimination of redundant reflections. This supports our broader finding that appropriate prompting strategies can mitigate the reflection overfitting observed in RLLMs, leading to more streamlined reasoning without sacrificing performance.

## D Additional Results

In addition to the MATH dataset analyzed in the main text, we further examined the distribution of thinking tokens on the relatively simpler ASDIV and GSM8K datasets. Figures 6 and 7 present the corresponding histograms of the number of thinking tokens under the three prompting methods: Direct, Zero-shot CoT, and Few-shot CoT.

Overall, we observe trends that are consistent with those identified on the more complex MATH dataset. First, Few-shot CoT generally yields a

more concentrated distribution of thinking tokens, mirroring the effect of example-based guidance seen in more challenging tasks. Meanwhile, Direct prompting tends to produce outputs that vary more widely in the number of thinking tokens, with a notable fraction of responses exhibiting very short or very long thinking parts. Interestingly, Zero-shot CoT prompts again lie between these two extremes, indicating that a brief, generic instruction to reason step by step partially constrains the model’s thinking process but does not standardize it as strongly as providing explicit exemplars.

Nevertheless, compared to MATH, the distributions on both ASDIV and GSM8K show that the majority of questions require fewer thinking tokens overall. This result aligns with the fact that these two datasets are simpler than MATH, which naturally leads to shorter solution paths and fewer opportunities for extensive reflections or self-corrections. We also note that, for larger-capacity models, the differences in thinking token distributions among correct and incorrect solutions are somewhat less pronounced than those observed in the MATH experiments, suggesting that complex tasks accentuate the benefits and nuances of prompting more sharply.

Despite these dataset-specific distinctions, the overarching pattern remains consistent: CoT prompting not only enhances the accuracy of reasoning LLMs but also regulates their reasoning length. In particular, the inclusion of even a short chain-of-thought instruction reduces the propensity for excessive self-reflection and focuses the models on more concise, goal-oriented reasoning steps.

## E Deadlock Reflection Pattern

We observe a severe failure mode in smaller RLLMs where the model enters a *deadlock reflection pattern*—repeatedly generating identical self-correction phrases without making progress. This pattern is particularly prevalent in R1-1.5B on complex AIME24 problems.

**Quantitative Analysis** On AIME24 with Direct prompting, R1-1.5B exhibits extreme reflection behavior: the average reflection count per instance is 838.2, with a maximum of 1,215 and minimum of 461 (std. dev. 125.7). A single problem can trigger over 800 occurrences of “wait” plus 200+ other reflective keywords.

**Example** The following excerpt shows R1-1.5B trapped in a deadlock pattern when solving an AIME24 geometry problem. The model repeatedly outputs identical statements without advancing its reasoning:

**Deadlock Pattern in R1-1.5B (Partial Output)**

*Wait*, no, that's not correct.  
*Wait*, actually, it's:  $g = \left[ -\frac{17}{(e-d)} - e - d \right] / 2$   
*Wait*, no, that's not correct.  
*Wait*, actually, it's:  $g = \left[ -\frac{17}{(e-d)} - e - d \right] / 2$   
*Wait*, no, that's not correct.  
[... pattern repeats 800+ times ...]

This deadlock behavior explains the extremely high reflection counts in Table 2 and demonstrates how smaller RLLMs can become trapped in unproductive self-correction loops when facing problems beyond their capacity. CoT prompting effectively prevents this pattern by providing structured guidance that anchors the reasoning process.

## F Large Language Model Usage

We employed Claude Sonnet 4 as a grammar expert to assist with proofreading this manuscript. Specifically, Claude Sonnet 4 was used solely to identify and correct linguistic issues including verb tense inconsistencies, grammatical errors, punctuation mistakes, and subordinate clause structures. The LLM's role was strictly limited to language polishing without any contribution to the research content, methodology, or scientific conclusions.

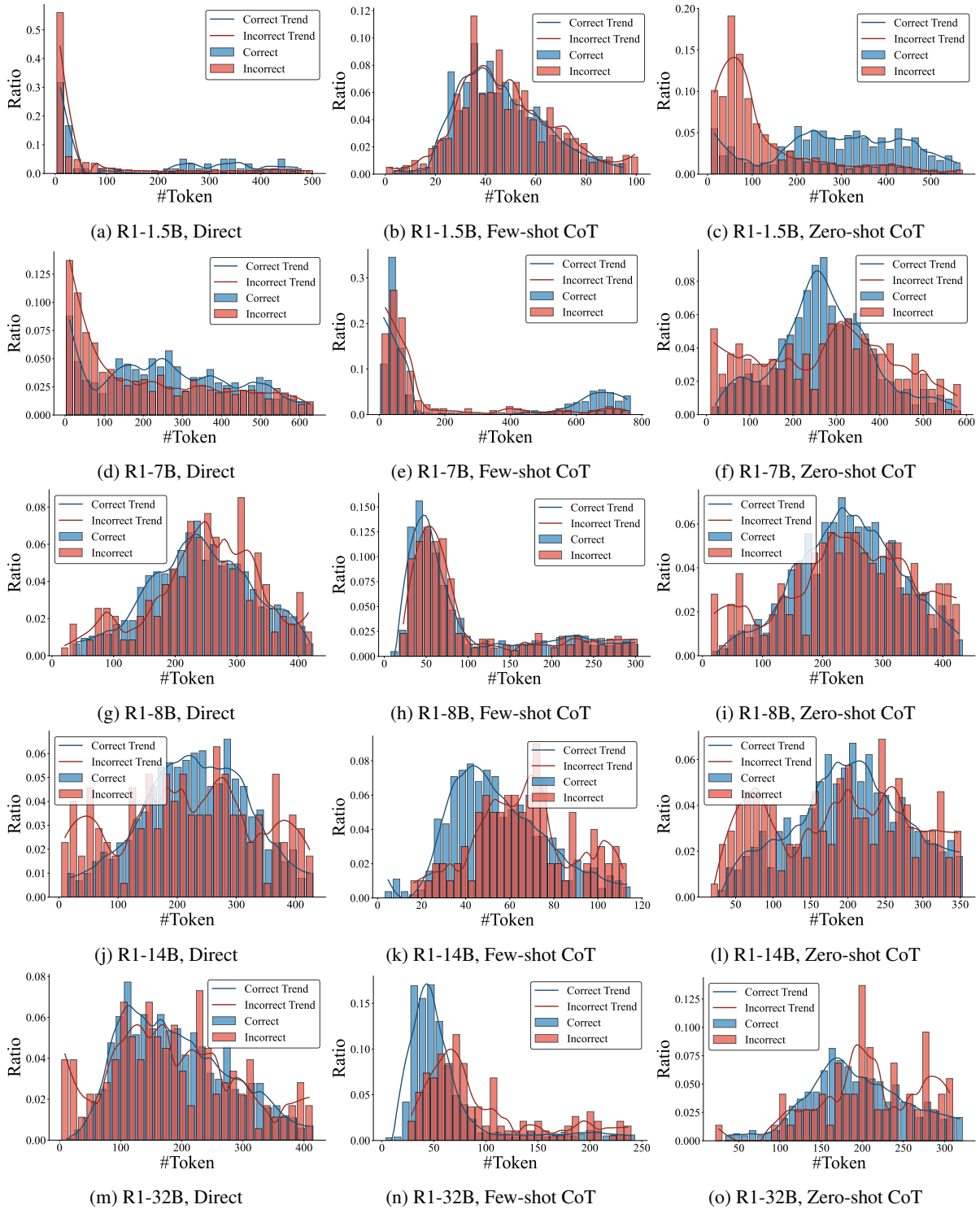


Figure 6: Distributions of thinking tokens across various RLLMs under three prompting methods evaluated on the GSM8K benchmark. The horizontal axis indicates the number of thinking tokens in the thinking parts (#Token), and the vertical axis represents the corresponding ratio. Histograms labeled “Correct” and “Incorrect” depict the distribution of token counts for correctly and incorrectly solved problems, respectively, while the trend lines (“Correct Trend” and “Incorrect Trend”) represent smoothed regression fits of these distributions.

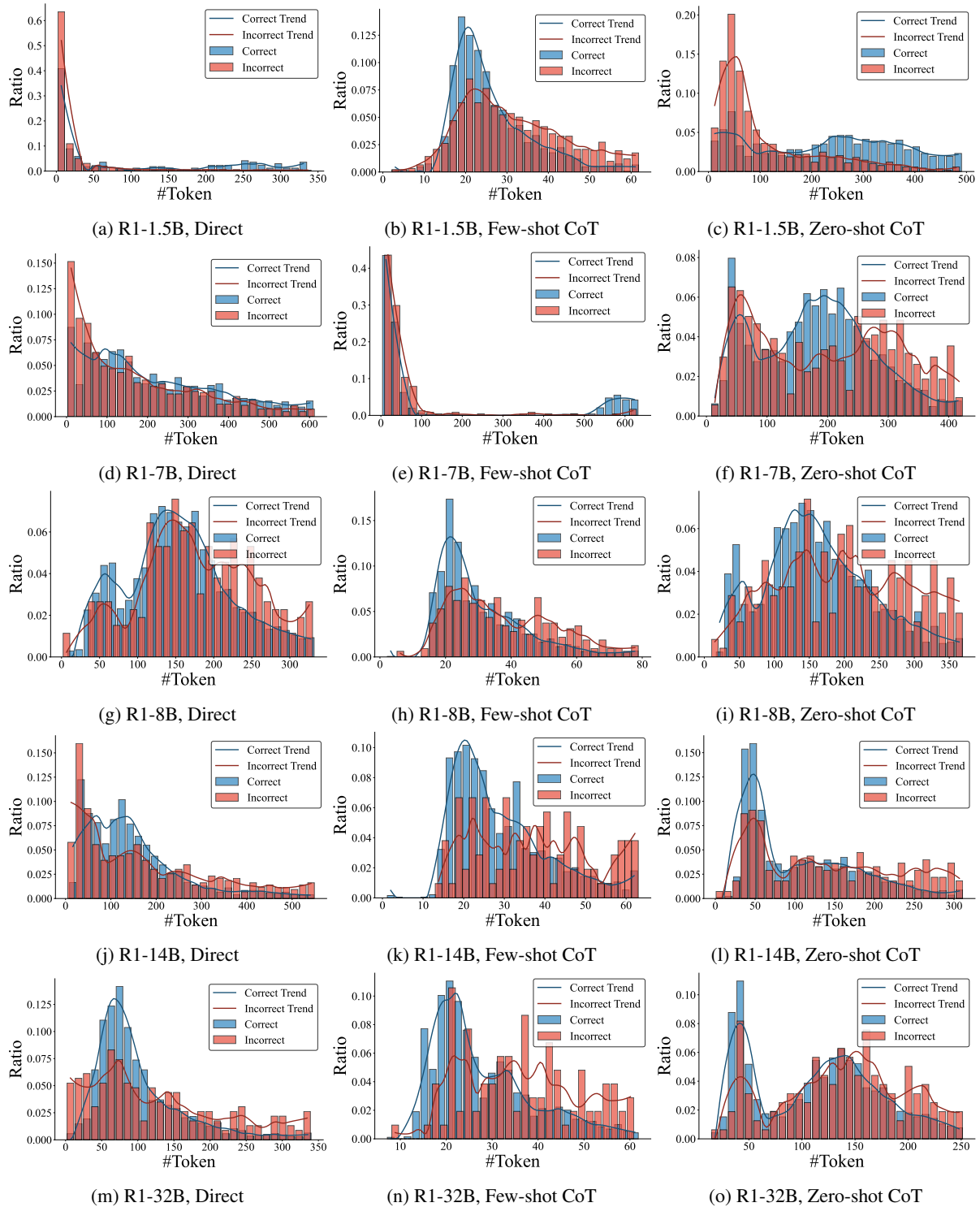


Figure 7: Distributions of thinking tokens across various RLLMs under three prompting methods evaluated on the ASDIV benchmark. The horizontal axis indicates the number of thinking tokens in the thinking parts (#Token), and the vertical axis represents the corresponding ratio. Histograms labeled “Correct” and “Incorrect” depict the distribution of token counts for correctly and incorrectly solved problems, respectively, while the trend lines (“Correct Trend” and “Incorrect Trend”) represent smoothed regression fits of these distributions.