Language Models Can Easily Learn to Reason from Demonstrations

Dacheng Li* UC Berkeley
Shiyi Cao* UC Berkeley
Tyler Griggs* UC Berkeley
Shu Liu* UC Berkeley
Xiangxi Mo UC Berkeley
Eric Tang Anyscale
Sumanth Hegde Anyscale
Kourosh Hakhamaneshi Anyscale
Shishir G. Patil UC Berkeley
Matei Zaharia UC Berkeley
Joseph E. Gonzalez UC Berkeley
Ion Stoica UC Berkeley

Abstract

Large reasoning models (LRMs) tackle complex problems by following long chain-of-thoughts (Long CoT) that incorporate reflection, backtracking, and self-validation. However, the training techniques and data requirements to elicit Long CoT remain poorly understood. In this work, we find that language models can effectively learn Long CoT reasoning through data-efficient supervised fine-tuning (SFT) and further parameter-efficient low-rank adaptation (LoRA). Crucially, we find that the structure of Long CoT is critical to the learning process in this data-efficient fine-tuning process. Training on content-incorrect examples, e.g. those lead to incorrect answers or corrupted digits, still leads to significant performance gains. In contrast, training on structurally incorrect examples, e.g., with shuffled or deleted reasoning steps, yield smaller improvements or even degrade performance. The model, Sky-T1, and codes are available at https://github.com/NovaSky-AI/SkyThought.

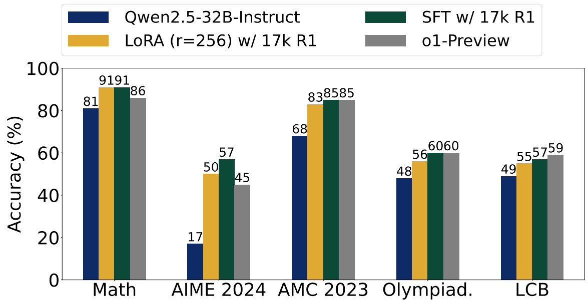

Figure 1: **Supervised fine-tuning (SFT) results of Qwen2.5-32B-Instruct**. When fine-tuned on a small amount (17k) of Long CoT samples reject-sampled from DeepSeek-R1 with either LoRA or full-parameter update, the model learns to perform reflection and backtracking by using keywords such as “However” and “Alternatively”. Consequently, the fine-tuned models improve significantly on five math and coding benchmarks (Olympiad. and LCB short for "OlympiadBench" and "LiveCodeBench"), matching OpenAI o1-preview performance. This shows that **learning to reason can be data-and parameter-efficient.**

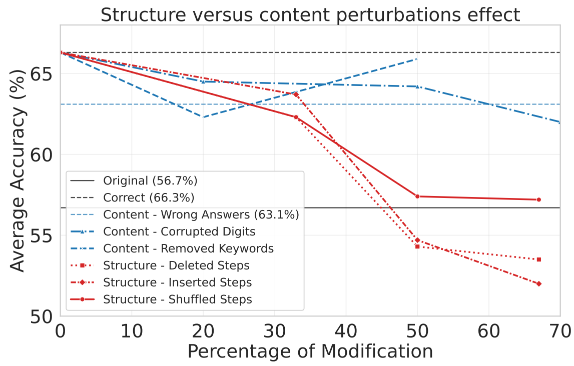

Figure 2: **SFT results on structure or content perturbed training data of Qwen2.5-32B-Instruct**. Performance is averaged over Math 500, AIME 24, AMC 23 and OlympiadBench. The construction procedure of these traces can be found at §4. We find that training on structurally perturbed traces (e.g., shuffled, inserted, or deleted steps) leads to significantly worse performance than training on content-perturbed traces (e.g., incorrect final answers, corrupted digits, or missing keywords). **This highlights the critical role of long CoT structure in learning to reason from demonstrations.**

1 Introduction

Large reasoning models (LRMs) leverage long chain-of-thoughts (Long CoTs) with reflection, backtracking, and self-validation to tackle challenging reasoning tasks (Jaech et al., 2024; Guo et al., 2025; Team, 2024). However, training LLMs to elicit Long CoTs remains an open problem, as existing methods are either closed-sourced (Jaech et al., 2024; Team, 2024) or expensive to replicate (Guo et al., 2025).

In this paper, we first show that an LLM can be easily taught to produce Long CoT responses, significantly improving its reasoning capabilities. In particular, we find that this learning process can be both data-efficient and parameter-efficient. By performing fully supervised fine-tuning (SFT)

*Equal contribution

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 15979–15997
November 4-9, 2025

with only 17k samples generated by DeepSeek R1, the Qwen2.5-32B-Instruct model achieves performance competitive with OpenAI o1-preview across a wide range of math and coding tasks (Team, 2024; Yang et al., 2024; Jaech et al., 2024). In particular, it achieves 90.8% in Math-500 (+6.0%), 56.7% in AIME 2024 (+40.0%), 85.0% in AMC 2023 (+17.5%), 60.3% in OlympiadBench (+12.7%) and 57.0% in LiveCodeBench (+8.1%) (Jain et al., 2024). Even further, the model can achieve o1-preview performance by updating fewer than 5% parameters with LoRA fine-tuning (Hu et al., 2021). We show that the model successfully learns to reflect and revise its intermediate thoughts (e.g., frequently using reasoning keywords such as "Alternatively" and "Wait, but") and adopts long, coherent CoTs to tackle challenging problems (Fig. 1).

Crucially, we identify the Long CoT *structure* as the key characteristic of distilled data for eliciting strong reasoning performance. We conduct two sets of controlled studies by altering either the content of individual reasoning steps or the overall logical structure. To alter content, we perturb samples by replacing numbers with random digits or deleting reasoning keywords. Surprisingly, we find that these perturbations have little impact on the model performance: even when 50% of numbers in training samples are randomly changed, the model only observes 3.3% lower accuracy on the most challenging math benchmark, AIME 2024, as compared to training with correct samples. To alter the long CoT structure, we separate responses into reasoning steps and randomly shuffle, insert, or delete these steps. We observe that the trained model is sensitive to such structural perturbations that break logical coherency in the long CoT. For example, when 67% of the training samples' reasoning steps are shuffled, accuracy drops by 13.3% on AIME 2024 problems relative to training with correct samples. Our key contributions are:

1. We demonstrate that an LLM can learn Long CoT reasoning in a data-efficient and parameter-efficient manner. With as few as 17k samples or less than 5% parameters update, we fine-tune the Qwen2.5-32B-Instruct model to be competitive with o1-preview.

2. We identify the structure of Long CoT as critical to the learning process rather than the content of individual reasoning steps. To validate this finding, we perform two groups of controlled experiments that modify either the structure or contents of samples.

3. We conduct comprehensive ablations across model sizes and architectures, dataset sizes, teacher models (DeepSeek R1 and QwQ-32B-Preview), and on five popular math and coding reasoning benchmarks.

2 Related work

Test Time Scaling for Large Language Models Scaling test-time compute has proven effective in enhancing the reasoning capabilities of LLMs. This can be broadly categorized into two directions: single long CoT and repeatedly sampled CoT. The former trains models, such as OpenAI o1, DeepSeek R1, and Qwen QwQ, to generate individual, long CoT responses with in-context reflection and backtracking to handle complex reasoning tasks (Guo et al., 2025; Jaech et al., 2024; Team, 2024). Alternatively, repeated sampling methods, such as Best-of-N or search-guided generation (e.g., MCTS), improve reasoning performance by invoking multiple responses from the model, sometimes guided by search algorithms and reward models (Snell et al., 2024; Brown et al., 2024; Li et al., 2025). In this paper, we focus on distilling the ability to generate individual, Long CoTs, and show it can be done in a data- and parameter-efficient manner.

Training to improve reasoning capabilities of LLMs LLM reasoning capabilities can be improved by approaches such as iterative self-improvement and reinforcement learning (RL) (Zelikman et al., 2022; Lightman et al., 2023; Lambert et al., 2024; Yuan et al., 2024; Guo et al., 2025). In particular, Setlur et al. (2024) shows that incorrect data can enable efficient reasoning capability training. More recently, Tulu-3 introduces Reinforcement Learning with Verifiable Rewards (RLVR) to improve performance in tasks such as math and coding (Hendrycks et al., 2021c; Jain et al., 2024; LI et al., 2024). PRIME proposes a RL-based method without process labels (Yuan et al., 2024). The recent release of DeepSeek R1 (Guo et al., 2025) demonstrates that LLMs can learn to produce long CoT and improve reasoning using a pure RL-based approach. Instead of bootstrapping reasoning ability, this paper focuses on the surprising data- and parameter-efficiency of distilling reasoning abilities from an existing reasoning model to an LLM. We find that training on incorrect data can also enhance LLM reasoning, but unlike (Setlur et al., 2024), we explore this in SFT instead of RL.

Distillation Distilling the outputs or logits generated by a larger or more capable model has become a standard technique to enhance model performance (Hinton, 2015). Typically, responses generated by higher-quality models are used to perform supervised fine-tuning on smaller models (Lambert et al., 2024). The Vicuna model, for instance, demonstrates that ChatGPT-generated responses can be used to effectively and cheaply distill high-quality chatting capabilities (Zheng et al., 2023). In this paper, we show that **reasoning capabilities can also be distilled**. We note that concurrent work has also observed similar trends in distilling reasoning capability (Min et al., 2024; Huang et al., 2024; Muennighoff et al., 2025; Ye et al., 2025). Our paper differs from these recent works by demonstrating that reasoning distillation can be achieved efficiently with minimal parameter updates, and the key role of the structure of long CoT in the process.

3 Experiments Setup

We describe the experiments settings for §4.

Distillation We use DeepSeek-R1 (Guo et al., 2025) and QwQ-32B-Preview (Team, 2024), two open-source models with reasoning capabilities, to generate our distillation data. We select difficult prompts from the AMC/AIME [1], Math, and Olympiad subset from the Numina-Math dataset (LI et al., 2024), as Min et al. (2024) implies that hard problems can improve performance. We also incorporate coding problems from APPS (Hendrycks et al., 2021a) and TACO (Li et al., 2023) datasets.

Specifically, we use GPT-4o-mini to classify the difficulty of the prompts according to the AoPS standard (Achiam et al., 2023), and select math problems of difficulty higher than Leval 3, Olympiad higher than Level 8, and all AIME/AMC problems. We verify the correctness of the traces by checking against ground truth solutions using exact matching for math problems and code execution for coding problems. In total, we curated 12k math and 5k coding problems with correct responses from QwQ to serve as our training data. For R1 samples, we directly use the public R1-17k reasoning dataset[2] that is curated following a similar procedure.

Training details We perform training using Llama-Factory (Zheng et al., 2024). We train the Qwen2.5-32B-Instruct using a batch size of 96, learning rate 1e-5 with a warm-up ratio of 0.1 and linear learning rate decay (Yang et al., 2024), following similar hyperparameters in (Min et al., 2024). We use the next token prediction loss as the training objective (Radford, 2018). We use the same hyper-parameters except a 1e-4 learning rate for LoRA fine-tuning. The full fine-tuning experiment is done with 8xH100 GPUs in 20 hours. Training is done with a single run.

Controlled experiments setup To understand the key factor in the learning process, we conducted controlled experiments in §4.1. In particular, we use QwQ-32B-Preview to produce the distillation data and select a subset of 4618 correct responses as the training set (out of the 12k math data above). The hyper-parameters remain the same as above.

Evaluation setup We evaluate on five popular reasoning benchmarks for math and coding, including Math-500, OlympiadBench, AIME-2024[3], AMC23[4] (Hendrycks et al., 2021c; He et al., 2024) and LiveCodeBench (Jain et al., 2024). For LiveCodeBench, we report a weighted average accuracy across the easy, medium, and hard difficulty levels.

4 Language Models Can Easily Learn to Reason From Demonstrations

In this section, we first show that a small amount of *well-curated* data, along with a simple parameter-efficient fine-tuning method (e.g., LoRA), can effectively improve reasoning capabilities in a large language model (§4.1). Then we present our observation that the long CoT structure is a key to the learning process (§4.2).

4.1 Learning from demonstration can be data-efficient and parameter-efficient

Small amount of data is enough. In Fig. 1, we present the performance of models fine-tuned with 17k R1 trained samples. Both the supervised fine-tuned (SFT) and LoRA fine-tuned models learn to generate Long CoT responses and improve significantly on all benchmarks with just 17k samples.

Less than 5% parameter update is enough. We next investigate whether full-parameter update is

[1]These prompts are from previous years of competition, which do not include AIME 2024 and AMC 2023 in our evaluation suite.

[2]huggingface.co/datasets/bespokelabs/Bespoke-Stratos-17k.

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

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

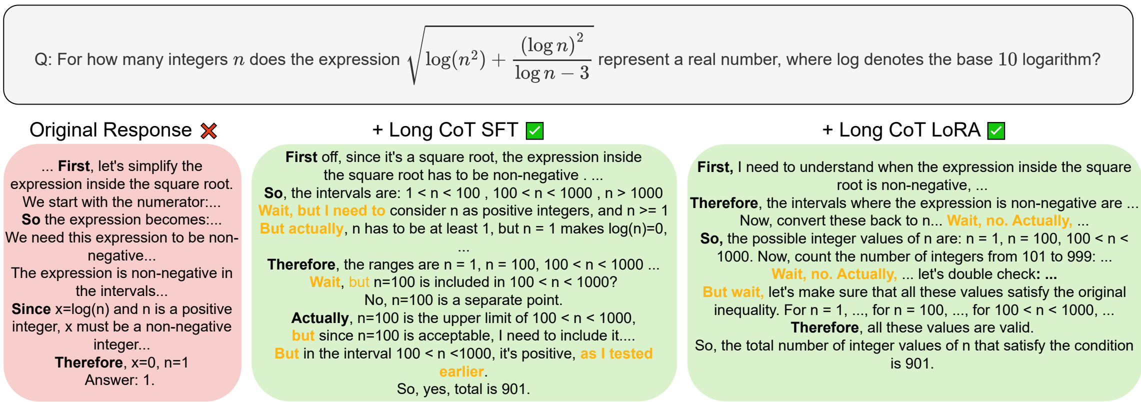

Figure 3: **Responses of the base model (Qwen2.5-32B-Instruct), with Long CoT full parameter or LoRA fine-tuning**. The model learns to reflect and backtrack using words like "Wait" and Actually" in both settings.

	MATH500	AIME24	AMC23	Olympiad.	LCB.
Qwen2.5-32B-Inst.	84.8	16.7	67.5	47.6	48.9
QwQ	90.4	33.3	75.0	58.1	59.1
o1-preview	85.5	44.6	87.5	59.2	59.1
7k QwQ Samples					
SFT	87.8	33.3	77.5	57.3	57.5
LoRA (r=64)	86.6	40.0	77.5	57.2	56.6
17k QwQ Samples					
SFT	87.8	33.3	70.0	56.7	57.9
LoRA (r=64)	86.6	33.3	90.0	56.0	56.2

Table 1: **Model accuracy with SFT and LoRA (rank=64).** Fine-tuning performed on Qwen2.5-32B-Instruct with QwQ samples. "Olympiad." is short for "OlympiadBench", "LCB." is short for "LiveCodeBench". We find that the learning process of Long CoT can be parameter efficient.

necessary. In addition to the results using 17k R1 samples as demonstrated in Fig. 1, we also report the results for both SFT and LoRA fine-tuning with 7k and 17k QwQ samples in Tab. 1.

Prior work (Ghosh et al., 2024; Biderman et al., 2024) suggests that LoRA fine-tuning substantially under-performs full fine-tuning for knowledge-intensive tasks, and is limited to learning response initiation and style tokens. However, our results in Fig. 1 and Tab. 1 show that LoRA fine-tuned models achieve similar or even superior reasoning performance compared to full-parameter SFT across math and coding benchmarks. Additionally, we find that a model fine-tuned with LoRA using just 7k QwQ samples performs comparably to one trained on 17k QwQ-distilled samples. This demonstrates that reasoning distillation can be achieved efficiently with both minimal parameter updates and minimal data. As shown in Fig. 3, the LoRA fine-tuned model easily learns to generate Long CoT responses with reflection and self-verification. These observations suggest that Long CoT reasoning ability may not rely on deep knowledge acquisition but rather on learning structured reasoning patterns, which can be effectively distilled in a parameter-efficient manner. This also aligns with prior findings that methods such as Chain-of-Thought prompting elicit Short CoT reasoning primarily by shaping response structure rather than instilling deep factual knowledge (Wei et al., 2022; Yao et al., 2023).

4.2 Global Long CoT reasoning structure is a key to the learning process

We further investigate the key factors in the fine-tuning process. Specifically, we consider two dimensions:

1. **The local content within a reasoning step**, including the correctness of the final answer, numbers in math derivations, and the use of reasoning keywords.

2. **The global reasoning structure**, including reflection, self-validation, and backtracking across multiple reasoning steps to form a logically coherent long CoT.

To understand their impact, we conduct two studies: (1) we perturb the content within individual reasoning steps – such as the final answer, numerical digits, and reasoning keywords(§4.2.1), and (2) we modify the global reasoning structure by inserting, deleting, and shuffling reasoning steps(§4.2.2). We compare the performance of models trained on perturbed samples against both the base Qwen2.5-32B-Instruct model (i.e., Original) and model trained on correct, unperturbed samples (i.e., Correct), as shown in Fig. 2. Our findings show that the learning process is highly sensitive to modifications in the long CoT reasoning structure, but remarkably tolerant to errors in the local contents.

4.2.1 Wrong or Corrupted Local Content

To study the importance of local content within individual steps, we preserve the overall reasoning structure while perturbing the local content in training samples with different approaches.

Wrong Answer Samples. During our training data curation process in §3, we only include samples that yield correct final answers. To assess whether correctness of the final answer is necessary for learning reasoning patterns, we instead train the model using an equivalent number of samples (4.6k) that lead to the *wrong* answer. Surprisingly, we find that training the base model **without any samples that reach a correct final answer** still achieves an average accuracy of 63.1% across benchmarks, only 3.2% lower than training with entirely correct samples.

Digits Corrupted Samples. Building on the previous experiment, we next examine the role of correctness in the intermediate reasoning steps. To evaluate this, we corrupt correct samples by replacing each digit with a random number between 0 and 9. Note that this is a severe corruption that can lead to nonsensical statements such as "1+1=3". Surprisingly, even when 70% of the digits are corrupted, the model still maintains an average performance of 62%, only 4.3% below the correct sample baseline, demonstrating robustness to incorrect content.

Reasoning Keyword Removal. Given the prevalence of reasoning *keywords* in responses from LRMs (e.g., "wait", "let me think again", "but"), one theory is that these specific phrases may invoke the reflection and back-tracking necessary to elicit strong reasoning performance. To evaluate it, we use GPT-4o-mini to identify sentences with occurrences of these reasoning keywords and randomly remove a fraction of them (e.g., 20%, 50%, 100%). Our results show that even after removing all (100%) such keywords, the model still achieves an average accuracy of 63%, which is within 3.3% of accuracy from the model trained with correct samples. This suggests that these particular keywords do not fundamentally impact the model reasoning performance.

Conclusion. The data-efficient fine-tuning process is robust to errors in local content – such as incorrect mathematical derivations or missing reasoning keywords.

4.2.2 Corrupted Global Reasoning Structure

Next, we examine the importance of reasoning *structure* by performing three modifications to the reasoning traces: deletion, insertion, and shuffle. We first note that our system prompt (Appendix C) instructs the model to generate responses with thoughts enclosed in the tags 'begin_of_thought' and 'end_of_thought' and the final solution and step-by-step explanation in 'begin_of_solution' and 'end_of_solution'. All modifications are performed on the *thoughts*, while the solution block is left unmodified.

We use Llama-3.3-70B-Instruct (Dubey et al., 2024) to separate each reasoning trace into distinct reasoning steps, with boundaries determined by occurrences of backtracking, self-validation, reflection, or other breaks from a linear sequence of thoughts. We then generated nine modified variants of the dataset by applying each modification (insertion, deletion, and shuffle – illustrated in Fig. 4) to 33%, 67%, or 100% of reasoning steps in the 4,618 correct traces. Each variant is used to train the base model, Qwen2.5-32B-Instruct, and we report the resulting performance in Fig. 2 and response lengths and reasoning keyword counts in Appendix D.

Deleted reasoning steps. As reasoning steps are increasingly deleted from the training data, model accuracy steadily declines and eventually regresses to the base model performance. Notably, retaining only the final solution and extensive step-by-step explanation (i.e., 100% deletion case) does not suffice to learn strong reasoning capabilities. This suggests that correct long CoT demonstrations alone are insufficient. Instead, examples of handling errors and dead ends with backtracking, reflection, and self-validation are important for eliciting robust reasoning.

At 67% deletion, the model imitates reasoning keywords (relative to the base model, keyword usage increases 45×, and output token increases 9×), but its accuracy does not improve accordingly. Consistent with §4.2.1, this validates that merely adopting reasoning keywords and long responses is insufficient. We note, however, that as more steps are deleted, the response lengths also decrease significantly, which could contribute to reduced accuracy. We hypothesize that it is the breaking of logical consistency *between* steps that causes accuracy degradation and validate this further in the following analysis.

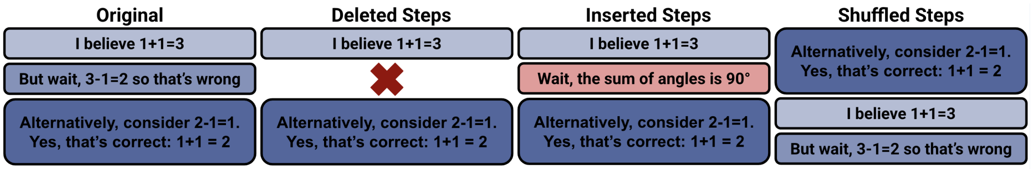

Figure 4: **Reasoning step modifications.** To evaluate perturbations to global structure across reasoning steps, we perform three modifications: deletion, insertion, and shuffling. These modifications break logical consistency across steps and degrade model accuracy far more than changes to local content within reasoning steps.

Inserted reasoning steps. To further validate the importance of logical structure, we replace a subset of each trace's reasoning steps with a random sample of reasoning steps from other samples in the training set that lead to correct results. Unlike deletion, this approach generally preserves the original length of the reasoning trace, ensuring that accuracy degradation is not due simply to producing fewer steps. Relative to model variants trained with deleted reasoning steps, variants trained on inserted steps generate longer responses with more reasoning keywords, yet accuracy deteriorates to, and even below, the level of the base model.

Interestingly, each inserted step is itself coherent and originates from a correct reasoning trace in the training data. Yet these internally-coherent steps appear in sequences that lack logical consistency and often from a separate domain (e.g., a combinatorics step may be inserted into a geometry solution), leading to contradictions and disjointed reflections. For example, the model trained with inserted reasoning steps frequently refers to earlier steps that do not exist (e.g., "Alternatively, consider a different approach" without a prior approach) or enumerates edge cases in inconsistent order (e.g., declaring a "Case 2" without "Case 1").

While the model readily produces coherent individual steps that reflect on a problem, the CoT fails to exhibit continuity *across* reasoning steps. This aligns with the observations in the deletion setting: a mere increase in reasoning steps or keywords is insufficient for robust reasoning—logical consistency across steps is a critical factor.

Shuffled reasoning steps. We next examine whether preserving the domain of each reasoning step, eliminating potential cross-domain confusion, but randomizing their order likewise impacts the model's ability to reason.

As the amount of shuffling increases, response length and reasoning keyword usage remain high, and in fact exceed the model trained on correctly ordered traces, yet accuracy declines sharply. Similar to the insertion experiments, the model imitates the syntax of per-step reasoning but loses logical consistency across steps. For instance, we find that over 92% of model responses begin with a backtracking or self-validation keyword (e.g., *"Alternatively," or "Wait"*), even though there is no preceding content to correct or reconsider. The model also references prior calculations or cases that were never actually introduced in any preceding step. Thus, while the shuffled traces still contain valid domain-specific reasoning steps, their rearrangement leads to incoherent overall solutions. In other words, domain alignment alone does not prevent logical breakdown.

Conclusions The correctness of the global long CoT structure is crucial in the data-efficient fine-tuning process.

5 Ablation Studies

In this section, we conduct a series of ablation studies to answer the following questions:

1. (§5.1) Does the use of more data improve the accuracy of the distillation?
2. (§5.2) Does fine-tuning on Long CoT data lead to degraded performance on non-reasoning tasks?
3. (§5.3) How much does the Long CoT fine-tuning enhance the performance of different student models?
4. (§5.4) How does the performance of the Long CoT fine-tuned model compare to the Best-of-N sampling performance of the base model?
5. (§5.5) How does Long CoT fine-tuning compare to Short CoT fine-tuning with the same dataset?

5.1 Data Scaling

We investigate the effect of distillation data size, ranging from 4k to 64k samples from R1, The results, presented in Fig. 5, show that a small amount

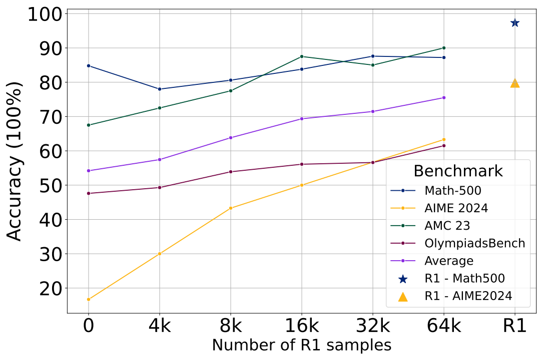

Figure 5: **Model accuracy with different data sizes, and comparison to DeepSeek R1.** The teacher model is DeepSeek R1, and the student model is Qwen-32B-Instruct trained with full parameter fine-tuning. While the student model continues to benefits from more SFT data from DeepSeek R1, a small amount of data, e.g., 16k is sufficient to significantly boost the average performance by 15.2%.

of data, e.g., 16k, is enough to significantly improve the model performance (from an average of 54.2 to 69.4).

5.2 Performance on Non-Reasoning Benchmarks

	MMLU	ARC-C	IEval	MGSM
Qwen2.5-32B-Inst.	74.1	49.4	78.7	42.3
QwQ	71.2	49.7	42.5	19.1
17k R1 Samples				
SFT	73.0	49.0	77.8	33.7
LoRA (r=256)	75.5	47.3	78.4	38.7
17k QwQ Samples				
SFT	78.4	49.5	75.8	33.0
LoRA (r=64)	78.5	46.7	74.1	30.6
7k QwQ Samples				
SFT	79.8	48.6	70.6	30.1
LoRA (r=64)	79.1	47.4	75.4	31.1

Table 2: **Distilled Model Performance on Non-Reasoning Tasks.** The teacher model is QwQ-32B-Preview, and the student model is Qwen2.5-32B-Instruct. Compared to QwQ, distilled models retain most of the base model's capabilities.

While simple distillation enhances reasoning capabilities, it is essential to ensure that these improvements do not come at the cost of catastrophic forgetting or a decline in general language understanding and instruction-following abilities, which are crucial for broader task generalization.

To assess this, we evaluate the performance of our SFT and LoRA fine-tuned models mentioned in §4.1 on a diverse set of benchmarks: MMLU (multi-task language understanding), ARC-C (science exam question), IEval (instruction-following), and MGSM (multilingual grade-school math problems) (Hendrycks et al., 2021b; Clark et al., 2018; Mitchell et al., 2023; Cobbe et al., 2021).

As shown in Tab. 2, the base instruction model (Qwen2.5-32B-Instruct) performs well in all these tasks. The QwQ model, despite its strong reasoning capabilities, suffers significant degradation in instruction-following (i.e., 42.5% on IEval) and multilingual tasks (i.e., 19.1% on MGSM). In contrast, fine-tuning (through both SFT and LoRA) only on a small amount of Long CoT reasoning data from R1 or QwQ allows the distilled models to retain most of the base instruction model's capabilities, avoiding the drastic performance drop seen in QwQ.

5.3 Effect on Different Student Models

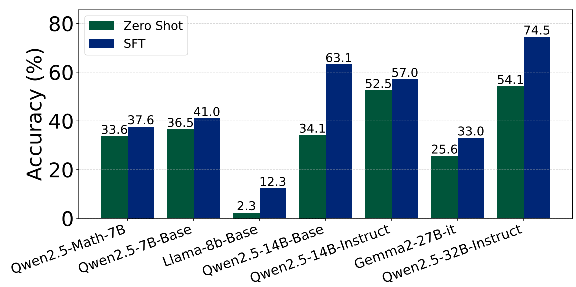

Figure 6: **Generalization to other models.** Averaged accuracy for different models with and without SFT. Models show significant improvements when fine-tuned with 17k samples from R1, showing that the Long CoT fine-tuning is beneficial across models.

In this section, we examine whether Long CoT reasoning capabilities can be elicited with different student models via fine-tuning (as described in §4.1). Specifically, we train with the 17k samples on Qwen2.5-7B-Math, Qwen2.5-7-Base, Llama-3.1-8B, Qwen2.5-14B-Base, Qwen2.5-14B-Instruct, Gemma2-27B-it and Qwen2.5-32B-Instruct (Yang et al., 2024; Dubey et al., 2024; Team et al., 2024). We find that these models improve noticeably across multiple benchmarks, showing the effect of Long CoT as a general improvement across models. However, not all models have showed the same degree of improvements as for Qwen2.5-32B-Instuct, suggesting a promising future direction for understanding the impact of various teacher and student models.

5.4 Comparison to Best-of-N

As observed in §5.3, not all student models achieve significant performance improvements through Long CoT fine-tuning. We hypothesize that this variation is influenced by several factors, such as the extent to which the training data distribution

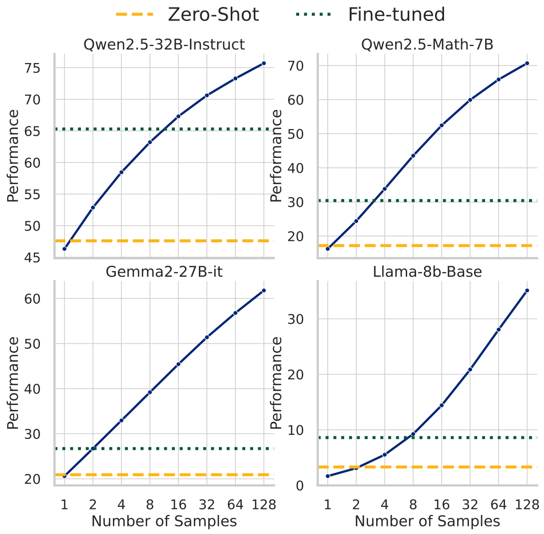

Figure 7: **SFT with Long CoT vs Best-of-N.** Accuracy of Qwen2.5-32B-Instruct before SFT (Zero-Shot), after SFT on 17k R1 samples (Fine-tuned), and Best-of-N samples on OlympiadBench. We find that fine-tuning on Long CoT achieves performance similar to Best of 2 to 16 samples.

differs from that of the student models and the inherent capabilities of the student models in these tasks. In this section, we compare the test-time scaling (Ahn et al., 2024; Snell et al., 2024) performance of the base model with its performance after Long CoT fine-tuning to understand the relationship between a model's ability to benefit from Long CoT fine-tuning and its intrinsic capabilities.

Specifically, we compare the performance of Long CoT fine-tuning against a Best-of-N sampling approach, where we generate 128 samples per prompt using an oracle verifier to select the best response. To introduce diversity, we employ a temperature of 0.5 and top-p sampling with a threshold of 0.8. The results, presented in Fig. 7, show that the Long CoT fine-tuned model performs comparably to Best-of-N sampling with 2 to 16 instances across all student models. Notably, the test-time scaling trends closely align with the improvements observed from Long CoT fine-tuning. For example, with eight parallel samples, Llama-3.1-8B achieves less than 10% accuracy on OlympiadBench, and similarly, fine-tuning with correct Long CoT traces results in only marginal improvement. A comparable trend is observed in Gemma2-27B-it and Qwen2.5-Math-7B, reinforcing the relationship between test-time sampling efficiency and the benefits of Long CoT fine-tuning.

The performance of Best-of-N sampling continues to improve beyond 128 samples, suggesting that further gains are possible. This highlights the potential for enhancing Long CoT models through context scaling or by leveraging a broader range of reasoning paths inherent to the original model, potentially unlocking even higher performance.

Dataset	Original	Short CoT	Long CoT
Avg. output tokens			
MATH500	684	515	3972
AMC23	728	605	5037
OlympiadBench	1275	948	8616
AIME24	825	687	15902
Avg. keywords per response			
MATH500	0.00	0.00	41.75
AMC23	0.00	0.00	39.20
OlympiadBench	0.01	0.01	97.20
AIME24	0.00	0.07	260.90
Performance			
MATH500	84.8	70.4 (-14.4)	**89.2** (+4.4)
AMC23	67.5	55.0 (-12.5)	**77.5** (+10.0)
OlympiadBench	47.6	36.4 (-11.2)	**58.5** (+10.9)
AIME24	16.7	13.3 (-3.4)	**40.0** (+23.3)

Table 3: **Comparison of number of output tokens reasoning keywords, and the performance between training with Short or Long CoT.** The original model is Qwen2.5-32B-Instruct. Benchmarks are ordered from easy to hard, where the model trained with Long CoT learns to produce longer CoTs and uses more keywords for harder problems.

5.5 Comparison to Short CoT Fine-tuning

In this section, we provide a direct comparison to training with short CoT. In particular, we compare results training on the 4.6k samples of Long CoT generated by QwQ-32B-Preview (§4.2), and the short CoT denoted in the NuminaMath-CoT dataset (LI et al., 2024). Tab. 3 summaries the statistics. Training with Long CoT enables the model to use more reasoning keywords (full list in Appendix B), produces longer responses to harder problems, and is the key to improved performance.

6 Conclusion

Large reasoning models unlock new capabilities by using a longer chain of thoughts that involves reflection and backtracking to answer challenging problems. In this paper, we show that such capability can be easily fine-tuned using a few thousand examples and with low-rank adapters. We further show that the long CoT structure of training examples is a key to the learning process. Finally, we discuss several ablations with various teacher-student models and compare them to the best-of-N approach.

Limitations

This work focuses on training reasoning models with supervised fine-tuning, while it does not compare the model behavior learned through Reinforcement Learning. Most conclusions for distillation are obtained with a small amount of data (e.g. 7k and 17k) , where it may not apply to large scale distillation as the one done in (Guo et al., 2025).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, and 1 others. 2024. Lora learns less and forgets less. *arXiv preprint arXiv:2405.09673*.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Chandler Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

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

Sreyan Ghosh, Chandra Kiran Reddy Evuru, Sonal Kumar, Ramaneswaran S, Deepali Aneja, Zeyu Jin, Ramani Duraiswami, and Dinesh Manocha. 2024. A closer look at the limitations of instruction tuning. *arXiv preprint arXiv:2402.05119*.

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*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *Preprint*, arXiv:2402.14008.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. 2021a. Measuring coding challenge competence with apps. *NeurIPS*.

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021c. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Geoffrey Hinton. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

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*.

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*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, and 1 others. 2024. T\" ulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.

Dacheng Li, Shiyi Cao, Chengkun Cao, Xiuyu Li, Shangyin Tan, Kurt Keutzer, Jiarong Xing, Joseph E Gonzalez, and Ion Stoica. 2025. S*: Test time scaling for code generation. *arXiv preprint arXiv:2502.14382*.

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. [https://huggingface.co/AI-MO/NuminaMath-CoT](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf).

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. 2023. Taco: Topics in algorithmic code generation dataset. *arXiv preprint arXiv:2312.14852*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Yingqian Min, Zhipeng Chen, Jinhao Jiang, Jie Chen, Jia Deng, Yiwen Hu, Yiru Tang, Jiapeng Wang, Xiaoxue Cheng, Huatong Song, and 1 others. 2024. Imitate, explore, and self-improve: A reproduction report on slow-thinking reasoning systems. *arXiv preprint arXiv:2412.09413*.

Eric Mitchell, Roberta Raileanu, Colin Raffel, John Levine, Yulia Tsvetkov, and Christopher D Manning. 2023. Ieval: An instruction following benchmark. *arXiv preprint arXiv:2310.07724*.

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*.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Amrith Setlur, Saurabh Garg, Xinyang Geng, Naman Garg, Virginia Smith, and Aviral Kumar. 2024. Rl on incorrect synthetic data scales the efficiency of llm math reasoning by eight-fold. *Advances in Neural Information Processing Systems*, 37:43000–43031.

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

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, and 1 others. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

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

Jason Wei and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

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*.

Shunyu Yao and 1 others. 2023. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*.

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*.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. 2024. Free process rewards without process labels. *arXiv preprint arXiv:2412.01981*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*.

A Short and Long CoT Response Examples

Question

For how many integers n does the expression

$$\sqrt{\frac{\log(n^2) - (\log n)^2}{\log n - 3}}$$

represent a real number, where $\log$ denotes the base 10 logarithm?

Short CoT answer (1/2)

To solve the problem, we start by analyzing the given expression and the conditions under which it represents a real number. The expression is:

$$\sqrt{\frac{\log(n^2) - (\log n)^2}{\log n - 3}}$$

For the square root to represent a real number, the argument of the square root, which is the fraction inside, must be nonnegative. Additionally, since we are dealing with logarithms, n must be a positive integer. Let's break down the solution into detailed steps:
1. Condition for the Denominator The denominator of the fraction is $\log n - 3$. For the fraction to be nonnegative, we need $\log n - 3 > 0$. This implies that $\log n > 3$, which translates to

$$n > 10^3 = 1000.$$

Therefore, n must be greater than 1000.
2. Simplifying the Numerator The numerator of the fraction is $\log(n^2) - (\log n)^2$. Using logarithmic properties, we can rewrite $\log(n^2)$ as $2 \log n$. Thus, the numerator becomes

$$2 \log n - (\log n)^2.$$

3. Analyzing the Fraction Substituting the simplified numerator into the fraction, we get:

$$\frac{2 \log n - (\log n)^2}{\log n - 3}$$

For this fraction to be nonnegative, given that the denominator is positive (from step 1), the numerator $2 \log n - (\log n)^2$ must also be nonnegative. This leads to the inequality:

$$2 \log n - (\log n)^2 \geq 0.$$

4. Solving the Inequality The inequality

$$2 \log n - (\log n)^2 \geq 0$$

can be rewritten as

$$(\log n)^2 - 2 \log n \leq 0.$$

Factoring,

$$\log n(\log n - 2) \leq 0.$$

Solving this inequality, we find that

$$0 \leq \log n \leq 2.$$

Converting back to n, this implies:

$$1 \leq n \leq 100.$$

Short CoT answer (2/2)

5. Combining Conditions From step 1, we have $n > 1000$, and from step 4, we have $1 \leq n \leq 100$. However, these two conditions are contradictory. The correct interpretation should be that n must satisfy both the condition of being greater than 1000 (from the denominator analysis) and the condition derived from the inequality, which was misinterpreted in the combination step. The correct combination should focus on the range derived from the inequality, which is $1 \leq n \leq 100$, and disregard the incorrect application of the denominator condition in this context. 6. Conclusion The integers n that satisfy the given conditions are those for which

$$1 \leq n \leq 100.$$

Counting these integers, we find there are

$$\boxed{100}$$

such integers.

Long CoT answer (1/4)

<|begin_of_thought|> So I've got this problem here:

$$\sqrt{\frac{\log(n^2) - (\log n)^2}{\log n - 3}}$$

And I need to find out for how many integer values of n this expression represents a real number. The logarithm here is base 10.
First off, since it's a square root, the expression inside the square root has to be non-negative for the whole thing to be a real number. So, that means:

$$\frac{\log(n^2) - (\log n)^2}{\log n - 3} \geq 0$$

Additionally, the argument of the logarithm has to be positive because logarithms are only defined for positive real numbers. So, $n^2 > 0$, which is true for all $n \neq 0$, and $n > 0$ since $n^2 > 0$ is always true for $n \neq 0$, but we need to ensure $n > 0$ because the logarithm of a negative number is undefined in real numbers.
So, n has to be a positive integer.
Now, let's simplify the expression inside the square root.
I know that $\log(n^2) = 2\log n$, because of the logarithm power rule. So, substituting that in:

$$\frac{2\log n - (\log n)^2}{\log n - 3}$$

Let me denote $\log n = x$ to make it easier. So, the expression becomes:

$$\frac{2x - x^2}{x - 3}$$

I can factor the numerator:

$$2x - x^2 = x(2 - x)$$

So, the expression is:

$$\frac{x(2 - x)}{x - 3}$$

I need to analyze when this fraction is non-negative.
To do that, I can look at the sign of each factor:
- $x = \log n$
- $2 - x = 2 - \log n$
- $x - 3 = \log n - 3$
So, the expression is:

$$\frac{\log n(2 - \log n)}{\log n - 3}$$

I need to determine when this fraction is greater than or equal to zero.
Let me consider the critical points where each factor is zero:

Long CoT answer (2/4)

1. $\log n = 0 \Rightarrow n = 10^0 = 1$
2. $2 - \log n = 0 \Rightarrow \log n = 2 \Rightarrow n = 10^2 = 100$
3. $\log n - 3 = 0 \Rightarrow \log n = 3 \Rightarrow n = 10^3 = 1000$

These points divide the number line into intervals where the expression doesn't change sign.
So, the intervals to consider are:
- $1 < n < 100$
- $100 < n < 1000$
- $n > 1000$

Wait, but I need to consider n as positive integers, and $n \geq 1$.
But actually, n has to be at least 1, but $n = 1$ makes $\log n = 0$, which is a critical point. So, let's consider the intervals:
- $1 < n < 100$
- $n = 100$
- $100 < n < 1000$
- $n = 1000$
- $n > 1000$

I need to test the sign of the expression in each interval.
First, let's consider $n = 1$:
$\log 1 = 0$, so the numerator is 0, and the denominator is $0 - 3 = -3$, so the fraction is $0/-3 = 0$, which is acceptable since it's zero.
Next, for $1 < n < 100$:
Let's pick $n = 10$, for example.
$\log 10 = 1$
So, $x = 1$
Then, $2 - x = 1$
$x - 3 = -2$
So, $\frac{1 \cdot 1}{-2} = -\frac{1}{2} < 0$, which is not acceptable.
Wait, but earlier I thought the expression inside the square root needs to be non-negative. So, this interval doesn't work.
Next interval: $n = 100$
$\log 100 = 2$
Numerator: $2 \cdot (2 - 2) = 0$
Denominator: $2 - 3 = -1$
So, $0/-1 = 0$, which is acceptable.
Then, $100 < n < 1000$
Let's pick $n = 1000^{0.5} = 10^{2.5} \approx 316.2$, but since n is integer, let's take $n = 300$
$\log 300 \approx 2.477$
So, $x \approx 2.477$
$2 - x \approx -0.477$
$x - 3 \approx -0.523$
So, $\frac{2.477 \cdot (-0.477)}{-0.523} \approx \frac{-1.182}{-0.523} \approx 2.26 > 0$, which is acceptable.
Next, $n = 1000$
$\log 1000 = 3$
Denominator: $3 - 3 = 0$, so the expression is undefined. Not acceptable.
Finally, $n > 1000$

Long CoT answer (3/4)

Let's pick $n = 10000$
$\log 10000 = 4$
So, $x = 4$
$2 - x = -2$
$x - 3 = 1$
So, $\frac{4 \cdot (-2)}{1} = -8 < 0$, which is not acceptable.
So, summarizing:
- $n = 1$: acceptable
- $1 < n < 100$: not acceptable
- $n = 100$: acceptable
- $100 < n < 1000$: acceptable
- $n = 1000$: not acceptable
- $n > 1000$: not acceptable
Therefore, the acceptable ranges are:
- $n = 1$
- $n = 100$
- $100 < n < 1000$
Wait, but $n = 100$ is included in $100 < n < 1000$? No, $n = 100$ is a separate point.
Actually, $n = 100$ is the upper limit of $100 < n < 1000$, but since $n = 100$ is acceptable, I need to include it.
Wait, no, $n = 100$ is a separate point where the expression is zero, which is acceptable.
So, the acceptable n are:
- $n = 1$
- $100 \leq n < 1000$
Wait, but earlier I thought $n = 1000$ is not acceptable, but $n = 1000$ is not included in $100 \leq n < 1000$, which is correct.
So, now, I need to count the integer values of n in these ranges.
First, $n = 1$: 1 value
Second, $100 \leq n < 1000$: that's from 100 to 999, inclusive.
So, $999 - 100 + 1 = 900$ values.
Therefore, total integers are $1 + 900 = 901$.
But wait, earlier I thought $n = 100$ is acceptable, and it is included in $100 \leq n < 1000$, so no need to count it separately.
Wait, no, $n = 100$ is included in $100 \leq n < 1000$, which already includes 100 to 999, so total is 900 values.
Plus $n = 1$, so total is 901.
But, I need to make sure that for all n in $100 \leq n < 1000$, the expression is positive.
Wait, except for $n = 100$, where it's zero, which is acceptable.
But in the interval $100 < n < 1000$, it's positive, as I tested earlier.
So, yes, total is 901.
Therefore, the number of integer values of n that satisfy the condition is 901. **Final Answer**

$$\boxed{901}$$

<|end_of_thought|>
<|begin_of_solution|>

Long CoT answer (4/4)

To determine for how many integer values of n the expression

$$\sqrt{\frac{\log(n^2) - (\log n)^2}{\log n - 3}}$$

represents a real number, we first note that the expression inside the square root must be non-negative:

$$\frac{\log(n^2) - (\log n)^2}{\log n - 3} \geq 0$$

Additionally, since logarithms are only defined for positive real numbers, we require $n > 0$.
1. Simplifying the expression
Using the logarithm power rule, we have:

$$\log(n^2) = 2 \log n$$

Thus, the expression becomes:

$$\frac{2 \log n - (\log n)^2}{\log n - 3}$$

Let $x = \log n$. Then, the expression simplifies to:

$$\frac{x(2 - x)}{x - 3}$$

2. Finding critical points
The critical points are where the numerator or denominator equals zero:
- $x = 0 \Rightarrow n = 1$ - $2 - x = 0 \Rightarrow x = 2 \Rightarrow n = 100$ - $x - 3 = 0 \Rightarrow x = 3 \Rightarrow n = 1000$
3. Analyzing intervals
The intervals to consider are:
- $1 < n < 100$ - $n = 100$ - $100 < n < 1000$ - $n = 1000$ - $n > 1000$
Testing the sign of the expression in each interval:
- For $n = 1$: The expression equals 0, which is acceptable. - For $1 < n < 100$: The expression is negative, not acceptable. - For $n = 100$: The expression equals 0, acceptable. - For $100 < n < 1000$: The expression is positive, acceptable. - For $n = 1000$: The expression is undefined, not acceptable. - For $n > 1000$: The expression is negative, not acceptable.
4. Counting acceptable integers
The acceptable ranges are:
- $n = 1$ - $100 \leq n < 1000$
The number of integers in $100 \leq n < 1000$ is 900 (from 100 to 999 inclusive). Adding $n = 1$ gives a total of:

$$1 + 900 = 901$$

Thus, the number of integer values of n that satisfy the condition is:

$$\boxed{901}$$

<|end_of_solution|>

B Reasoning keywords

We create a bank of reasoning keywords by manual inspection of reasoning responses from QwQ and R1. We identify these following words and phrases as frequent indicators of reflection, backtracking, and self-validation: “Alternatively”, “Wait”, “Just to be thorough”, “Just to make sure”, “Let me just double-check”, “Let me try another”, “Let me verify”, “Let me check”, “Hmm”, “But”, “Maybe I should consider”, “Maybe I can consider”.

C Long CoT System Prompt

We adopt the following system prompt from Still-2 (Min et al., 2024):

Long CoT System Prompt

> Your role as an assistant involves thoroughly exploring questions through a systematic long thinking process before providing the final precise and accurate solutions. This requires engaging in a comprehensive cycle of analysis, summarizing, exploration, reassessment, reflection, backtracking, and iteration to develop well-considered thinking process. Please structure your response into two main sections: Thought and Solution. In the Thought section, detail your reasoning process using the specified format: <|begin_of_thought|> thought with steps separated with `\n\n}` <|end_of_thought|> Each step should include detailed considerations such as analyzing questions, summarizing relevant findings, brainstorming new ideas, verifying the accuracy of the current steps, refining any errors, and revisiting previous steps. In the Solution section, based on various attempts, explorations, and reflections from the Thought section, systematically present the final solution that you deem correct. The solution should remain a logical, accurate, concise expression style and detail necessary step needed to reach the conclusion, formatted as follows: <|begin_of_solution|> final formatted, precise, and clear solution <|end_of_solution|> Now, try to solve the following question through the above guidelines:

D Average response lengths and keyword counts

Table 4: Average keyword counts and output tokens for deleted steps.

Dataset	**0%**	**33%**	**67%**	**100%**
Avg. output tokens				
Math	3551	2979	2078	482
AMC 2023	4838	6612	4623	609
OlympiadBench	7234	6802	4978	595
AIME 2024	13088	11889	6798	620
Avg. keywords per response				
Math	32	28	20	0.017
AMC 2023	39	85.6	77.8	0
OlympiadBench	77	70	56	0.009
AIME 2024	143	143	90	0

Table 5: Average keyword counts and output tokens for inserted steps.

Dataset	**0%**	**33%**	**67%**	**100%**
Avg. output tokens				
Math	3551	4189	3900	5383
AMC 2023	4838	7089	5464	5137
OlympiadBench	7234	7558	6990	5407
AIME 2024	13088	12858	12864	5304
Avg. keywords per response				
Math	32	39	39	41
AMC 2023	39	98	44	35
OlympiadBench	77	76	80	38
AIME 2024	143	127	165	44

Table 6: Average keyword counts and output tokens for shuffled steps.

Dataset	**0%**	**33%**	**67%**	**100%**
Avg. output tokens				
Math	3551	4284	5784	5613
AMC 2023	4838	6802	10198	8661
OlympiadBench	7234	8942	12154	12167
AIME 2024	13088	13451	16221	18054
Avg. keywords per response				
Math	32	45	61	70
AMC 2023	39	65	74	67
OlympiadBench	77	111	166	137
AIME 2024	143	161	201	210