MORALE: Segment-Guided Distillation Framework for Small Reasoning Models

Anonymous ACL submission

Abstract

Chain-of-Thought (CoT) significantly enhances Large Language Models (LLMs) reasoning, but distilling complex CoT to small models remains challenging. Naive distillation often yields limited gains or even degrades performance in small models, likely due to their capacity constraints. Existing methods improving small model reasoning mainly rely on costly and impractical ground truth answers for data selection. Moreover, they lack a trade-off between performance and output length. To address this, we propose **MORALE**, a seg**M**ent-guided distillati**O**n framewo**R**k for sm**AL**l r**E**asoning models. MORALE enables small models to learn from complex CoT knowledge effectively, without requiring ground truth verification, maintaining high performance with remarkably short outputs. Specifically, MORALE divides reasoning trajectories into independent segments complemented by a summary, making it *dataset-agnostic*. Then, we propose a two-stage rejection sampling with direct preference optimization to further boost model potential while keeping thinking concise. Extensive experiments demonstrate that MORALE substantially improves small model reasoning performance, achieving an average gain of 36.93%, while simultaneously reducing output length by 65.86% compared to conventional long CoT distillation.

1 Introduction

Recent Large Language Models (LLMs) (Touvron et al., 2023; Bai et al., 2023; Achiam et al., 2023; Team et al., 2024; Liu et al., 2024) have achieved remarkable performance, demonstrating significant advancements across various domains, including natural language understanding, machine translation, and question-answering systems. Particularly notable is the progress in complex reasoning. Techniques such as Chain-of-Thought (CoT) prompting (Wei et al., 2022; Feng et al., 2023) are instrumental in enabling LLMs to substantially improve their reasoning capabilities, often applied at inference stage. As an illustration, models such as OpenAI's o1 (Jaech et al., 2024), Deepseek R1 (Guo et al., 2025), and Kimi k1.5 (Team et al., 2025) have achieved unprecedented performance in areas like mathematics, code generation, and logical reasoning tasks.

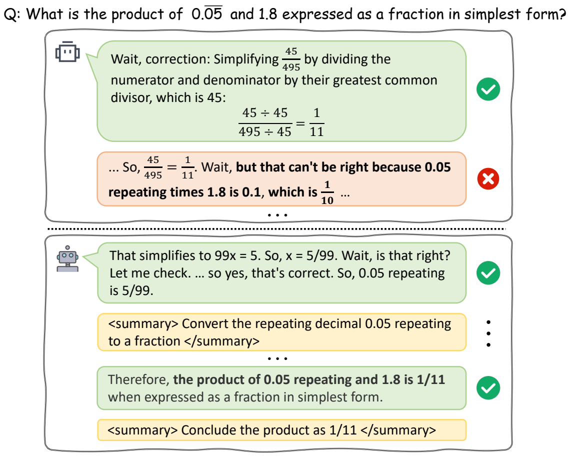


Figure 1: Top: Small models become logically confused in the intermediate steps of conventional long CoT, negating their correct answer and instead starting to explore incorrect paths. Bottom: MORALE enables small models to perform a summary after each independent segment, while clarifying the problem-solving reasoning and maintaining consistency.

The advanced reasoning capacity is not free due to the substantial resource consumption and inference latency, necessitating the development of cost-effective alternatives (Ranaldi and Freitas, 2024; Shridhar et al., 2022; Wang et al., 2025a). Knowledge distillation from LRMs is a common practice to develop such models, aiming to replicate their advanced reasoning and self-correction capabilities (Kim et al., 2024; Hinton et al., 2015; Agarwal et al., 2024). However, conventional distillation often yields suboptimal results (Fu et al., 2023; Zhang et al., 2025; Shridhar et al., 2023), including repetitive outputs or logical inconsistencies, likely attributed to the inherent limitation of small model size[1]. While studies like (Luo et al., 2025; Li et al., 2025) explore improving the reasoning ability of small models through distillation of complex and multi-step reasoning (namely long CoT), they face a significant challenge: ground-truth supervision is expensive and labor intensive. Such reliance imposes stringent requirements on long CoT datasets and necessitates discarding valuable data (Yang et al., 2024b; Yan et al., 2025). This data bottleneck, coupled with findings suggesting the importance of maintaining the teacher's detailed reasoning structure: for example, Luo et al. (2025) observed that removing detailed or even incorrect reasoning steps does not significantly benefit distillation outcomes, underscores the inherent difficulty in effectively transferring complex reasoning into small models. In a nutshell, how to optimize the long CoT distillation process by retaining both the short test-time reasoning and promising performance for small models is an open question.

To address the aforementioned challenge, we propose MORALE, a simple yet effective approach that enables small models to fully utilize the potential of long CoT. The core premise is that long CoT reasoning involves intricate formulas and logical sequences that are difficult for small models to process. This difficulty can lead to errors, sometimes resulting in correct answer negating, as illustrated in Figure 1. Here, we propose to leverage a sufficiently capable LLM to *segment intermediate reasoning steps and perform semantic summarization for the formers*, thereby facilitating knowledge transfer to small models. Specifically, our method can be divided into three main steps: 1) The first step involves equipping a powerful LLM with the capability to split long CoT into logically coherent segments and provide a high-level summary of these segments. 2) The second step involves applying the above LLM to segment raw long CoT SFT data (e.g., the outputs from LRMs like R1) and injecting these high-level summaries as auxiliary information. 3) The Two-stage Rejection Sampling (RS) with Direct Preference Optimization (DPO) (namely RS2DPO) module utilizes the model's inherent capabilities to enhance the generation of correct and concise solution paths, thereby mitigating the tendency for "overthinking".

Our experiments demonstrate that, under the same training configuration, our method outperforms conventional long CoT distillation on several open-source long CoT datasets. For instance, on the OpenR1-5K dataset, the Qwen2.5-Math-1.5B model trained with our method achieved an average performance improvement of 2.58%, with significant gain observed in both in-domain and out-of-domain settings. Moreover, consistent performance gains were observed across different models, datasets, and evaluation benchmarks. Furthermore, applying the RS2DPO paradigm enhances model performance and reduces the output length.

In summary, the contributions of this work are as follows:

- We highlight that long CoT could hurt the distillation of the small models due to the complex and intricate reasoning paths, which are hard to learn by the former.
- We propose MORALE, a novel method that enhances small models' acquisition of long CoT knowledge by transforming complex reasoning steps into concise, generalized segments and their summaries. Furthermore, we propose a two-stage rejection sampling with DPO for model reasoning preference alignment, which sufficiently improves reasoning performance with the short paths.
- We conducted extensive experiments to validate the effectiveness and generality of our method across different model sizes and datasets.

2 Related Work

Reasoning Space Exploration Increasing the inference-time token budget has proven highly effective in extending reasoning length and improving performance, exemplified by OpenAI's o1 series and DeepSeek R1. Test-time scaling typically employs two main paradigms: **(i) Parallel scaling** involves generating multiple trials in parallel to enhance reasoning. Methods like ToT(Yao et al., 2023) enable LLMs to explore diverse reasoning paths towards problem-solving. However, other studies(Parashar et al., 2025; Wang et al., 2025b; Zheng et al., 2024) question the actual effectiveness and performance gains from simply increasing inference time. **(ii) Vertical scaling** incorporates substructures for more fine-grained reasoning, such as problem restatement, approach analysis, and step verification(Luo et al., 2025; He et al., 2025). State-of-the-art methods, notably R1(Guo et al., 2025),

[1]A bad case example is depicted in Figure 7 in the Appendix

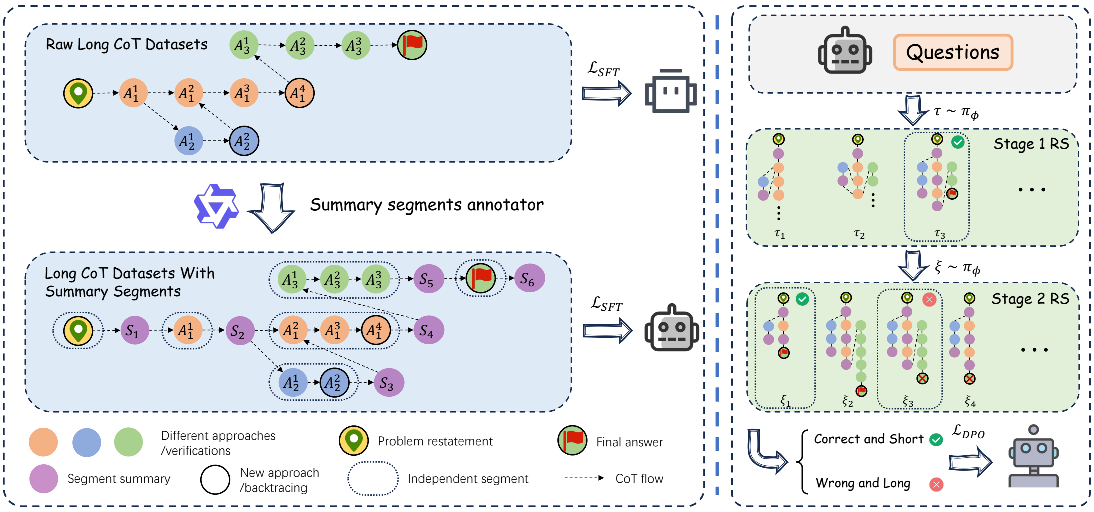


Figure 2: An Overview of MORALE. The left part illustrates the process of constructing self-contained segments along with corresponding summary, and the right part depicts the Two-stage-RS with DPO pipeline, denoted as RS2DPO.

leverage reinforcement learning to guide models in developing human-like strategies, including self-doubt and reflection.

Knowledge Distillation While reinforcement learning significantly enhances reasoning, its high resource requirements and instability pose challenges. For small models, distillation from LRMs(Huang et al., 2024) offers a simpler alternative. Research indicates that distilled models acquire diverse reasoning features, including self-reflection and computation verification(Baek and Tegmark, 2025). To address issues like overthinking and learning difficulty in distillation, methods like constructing tree-based CoT data via Monte Carlo Tree Search (MCTS) have been explored(Yin et al., 2025). Nevertheless, recent studies underscore that distilling intricate and extensive chains of thought from LRMs into small models remains a significant challenge(Li et al., 2025; Wang et al., 2025c; Yang et al., 2025; Liu et al., 2025; Sui et al., 2025).

Overthinking Phenomenon Some studies have shown that forcibly transferring LRM reasoning capabilities to small models can lead to logical inconsistencies and redundant steps(Chen et al., 2024b; Pan et al., 2024; Cuadron et al., 2025; Kumar et al., 2025). While reasoning performance is often believed to be proportional to length, research suggests removing natural language thought descriptions has minimal impact on Small Reasoning Models (SRMs) efficacy(Ma et al., 2025). Furthermore, analyses indicate that LLM reasoning collapses beyond certain boundaries, resulting in abnormal output(Chen et al., 2024a). The phenomenon of overthinking has also been likened to a "snowball error," where the model output errors accumulate sequentially(Gan et al., 2025).

3 Methodology

3.1 Preliminaries

Notations Let $x = (x_1, x_2, \ldots, x_n)$ represent an input sequence, $y = (y_1, y_2, \ldots, y_m)$ represent corresponding output sequence. An LLM parameterized by θ will predict the next token at step t according to a conditional distribution $\pi_\theta(y_t|x, y_{1:t-1})$. Generally, the output sequence can be divided into a series of independent reasoning steps: $\eta = (r_1, r_2, \ldots, r_k, a)$, where η is a completed reasoning trajectory and a denotes the final answer, while each r_i is a complete atomic problem-solving process or an attempt and is always composed of $\{y_i\}_p^q$.

Supervised fine-tuning (SFT) is widely adopted to enhance reasoning ability of LLMs on a dataset $\mathcal{D} = \{x_i, \eta_i\}_{i=1}^N$, where η_i is generated from a powerful LRM and conssists of lengthy verification and backtracing. The SFT process distill the complete reasoning paradigm from LRM to small models through updating the parameter θ of a LLM by minimization the negative log-likelihood loss over the long CoT dataset $\mathcal{D}_{\text{SFT}}$.

$$\mathcal{L}_{\text{SFT}}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K_i} \log P_\theta(a_i|x_i, r_{i,1}, \ldots, r_{i,k}) \tag{1}$$

Direct preference optimization (DPO) (Rafailov et al., 2023) is employed to align LLMs with human preferences by promoting concise and accurate trajectory η^w while discouraging verbose and repetitive one η^l, all within the constraints imposed by a reference policy π_{ref} on a dataset $\mathcal{D}_{\text{DPO}} = \{(x_i, \eta_i^w, \eta_i^l)\}_{i=1}^M$. This optimization framework ensures controlled model enhancement, improving the generation of high-quality outputs without deviating significantly from desired behavior.

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,\eta^w,\eta^l)\sim\mathcal{D}_{\text{DPO}}} \left[\log\sigma\left(\beta\log\frac{\pi_\theta(\eta^w \mid x)}{\pi_{\text{ref}}(\eta^w \mid x)} - \beta\log\frac{\pi_\theta(\eta^l \mid x)}{\pi_{\text{ref}}(\eta^l \mid x)}\right)\right] \tag{2}$$

3.2 MORALE: Make Full Use of Long CoT

The main objectives of MORALE are diveded into two parts: (1) improve the model's learning efficiency on long CoT data (§3.2.2); (2) unleash the model's intrinsic potential (§3.2.3). Figure 2 presents the overall framework of MORALE.

3.2.1 Segmentation Preparation

Our method begins by requiring a sufficiently powerful LLM capable of identifying independent segments within a long CoT. These independent segments include atomic operations such as problem-solving steps and reflections. Traditional CoT decomposes solutions into steps, but long CoT contains a large number of intermediate steps, many of which are overly granular and do not significantly contribute to the overall reasoning. Inspired by DeltaBench(He et al., 2025), which provids a batch of segmentations annotated by human experts on long CoT, covering 5 major domains and 48 subcategories, we finetuned a specialized expert LLM using this open-source dataset to segment long CoT data and concurrently extract the corresponding summary. Specific details are provided in Appendix A.1.

3.2.2 Dataset Curation

Given an initial long CoT dataset:

$$\mathcal{D} = \{(x_i, y_{i,1}, y_{i,2}, \ldots, y_{i,n})\}_{i=1}^N$$

generated by a powerful LRM(e.g., R1), we first segment the dataset into logically independent units using a prepared segmentation model, π_{seg}. This transformation yields:

$$\mathcal{D}_{\text{seg}} = \{(x_i, r_{i,1}, r_{i,2}, \ldots, r_{i,m}, a)\}_{i=1}^N$$

Here, $r_{i,j} := \{y_{i,p}, y_{i,p+1}, \ldots, y_{i,q-1}, y_{i,q}\}$ represents a self-contained segment, typically corresponding to a complete atomic problem-solving process or a distinct solution attempt.

To facilitate efficient knowledge distillation and enhance clarity for small models, we append a concise summary to each segment. The resulting dataset is thus:

$$\mathcal{D}_{\text{sum}} = \{(x_i, r_{i,1}, s_{i,1}, \ldots, r_{i,m}, s_{i,m}, a)\}_{i=1}^N$$

where $s_{i,j}$ is a summary of the preceding segment, designed to improve memorization and comprehension. Data example can be found in Figure 11 of Appendix C.1.

Afterwards, this constructed dataset are used for the model's SFT training, producing model π_ϕ. long CoT data incorporating segment summaries can smooth the learning curve and effectively enhance the reasoning capabilities of small models.

3.2.3 Two-stage RS with DPO (RS2DPO)

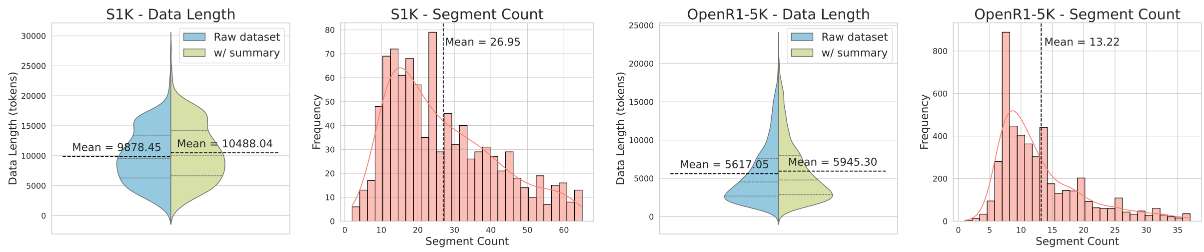
Our method has equipped the small model with superior efficacy in learning long CoT knowledge on the same dataset, leading to enhanced performance. However, as noted previously (§2), the model is susceptible to overthinking, characterized by excessive length and repetition in outputs. To mitigate this issue, fully leverage the model's capabilities, and minimize repetitive outputs, we employ a two-stage RS process. This process is designed to generate chosen and rejected data for subsequent DPO.

Stage 1 RS The first stage initiates RS on a batch of reasoning problems using the previously distilled model π_ϕ. This process yields M_{stage1} candidate outputs for each problem, represented as:

$$\mathcal{T} = \{\tau_i\}_{i=1}^{M_{\text{stage1}}}, \quad \tau_i \sim \pi_\phi$$

where each τ_i is sampled from the distilled model π_ϕ. While segment-guided distillation mitigates the difficulty of learning, many of these initial outputs can still be lengthy. From the generated pool $\mathcal{T}$, we identify and select outputs that successfully reach the correct answer without exhibiting excessive length or repetition.

Stage 2 RS In the second stage, the final segments of the selected trajectories (the thinking process) are truncated by removing their final k segments. Specifically, for a complete trajectory $\tau_i = \{r_{i,1}, s_{i,1}, r_{i,2}, s_{i,2}, \ldots, r_{i,l}, s_{i,l}, a\}$, the truncated sequence $\tau_i' = \{r_{i,1}, s_{i,1}, \ldots, r_{i,l-k}, s_{i,l-k}\}$



(a) (b) (c) (d)

Figure 3: Detailed analysis of the datasets we used. Figures (a) and (c) illustrate the distinction between the "Raw dataset" (the original open-source dataset) and the dataset "w/ summary" (referring to the addition of summary segments). Figures (b) and (d) present the resulting segment counts.

is used as input. RS is then re-applied using these truncated sequences as prompts to generate a new set of M_{stage2} candidate outputs:

$$\mathcal{X} = \{\xi_i\}_{i=1}^{M_{\text{stage2}}}, \quad \xi_i \sim \pi_\phi$$

where each ξ_i is also sampled from the distilled model π_ϕ. From this second set $\mathcal{X}$, correct and concise derivations are selected as chosen data, while incorrect or verbose derivations are selected as rejected data.

Finally, DPO training is conducted using this paired chosen and rejected data, producing the final model π_ψ. This entire framework is engineered to cultivate the model's proficient reasoning abilities and reinforce the discovery of accurate and efficient solution pathways within the problem-solving space.

4 Experiments

4.1 Experiment Settings

Benchmarks To verify the effectiveness and generality of the proposed method, we conducted evaluations on a wide range of mathematical and logical reasoning datasets. In addition to the widely used GSM8K(Cobbe et al., 2021), we incorporated challenging benchmarks from multiple domains, including: *(i) in-domain* competition and Olympiad-level benchmarks, such as MATH(Hendrycks et al., 2021), AIME 2024(AI-MO, 2024), and Minerva-Math(Lewkowycz et al., 2022). Specifically, AIME is a challenging mathematics competition designed for high school students in the United States; *(ii) out-of-domain* logical reasoning benchmarks: BBH(Suzgun et al., 2023) and DROP(Dua et al., 2019), focusing on assessing the ability of language models to perform complex reasoning tasks that go beyond simple pattern recognition or surface-level understanding.

Baselines Since our method is designed to fully leverage the potential of long CoT, we will use models fine-tuned on the same datasets as the baseline for our experiments. We compare the performance difference between our trained model and the baseline model on several reasoning benchmarks. All our SFT experiments use the same configuration and use LLaMA-Factory[2] as our training framework. More details are listed in Appendix A .

Models and Datasets To demonstrate the generality of our method, we used models of different sizes and datasets from different sources.

For our experiments, we selected models from the `Qwen2.5-Math-Instruct` series (Yang et al., 2024a) and the `Deepseek-Math` series (Shao et al., 2024). These series are commonly used in related research, following previous work (Guo et al., 2025; Guan et al., 2025). Specifically, we utilized `Qwen2.5-Math-1.5B-Instruct`, `Qwen2.5-Math-7B-Instruct`, and `Deepseek-Math-7B-Instruct`. For brevity, these models will hereafter be referred to as `Qwen-1.5B`, `Qwen-7B`, and `Deepseek-7B`, respectively.

For our experiments, we used several publicly available distillation datasets generated by the R1 reasoning model. These datasets, covering a range of sizes, include `S1K` (Muennighoff et al., 2025) and `OpenR1-5k`[3] (Face, 2025). Figure 3 illustrates the distribution of data length and segment counts for these datasets. Additionally, we used a larger dataset `Bespoke-Stratos-17K`, with further details available in Table 4 of Appendix B.

Evaluation Metrics For all the benchmarks we used in experiments, every test sample contains a question and a golden answer. We extract the final answer from the model's output and matched it with the golden answer to ultimately

[2] https://github.com/hiyouga/LLaMA-Factory
[3] Sampled randomly from OpenR1

Table 1: Our main experimental results on four mathematical reasoning tasks(GSM8K, MATH, Minerva-Math and AIME24) under *pass*@1 setting. We present both the absolute scores of our method and long CoT across different datasets and models and the relative improvement compared to vanilla models. Here, "+" indicates a score higher than the vanilla, whereas "-" indicates a lower score. Finally, average scores (abbreviated as "Avg.") are also reported.

Models	Method	GSM8K	MATH	Minerva-Math	AIME24	Avg.
Qwen2.5-1.5B	Vanilla	80.29	74.86	20.96	6.67	45.70
Qwen2.5-7B	Vanilla	73.84	81.70	29.41	16.67	50.41
Deepseek-7B	Vanilla	80.29	45.74	15.44	6.67	37.04
S1K						
Qwen2.5-1.5B	Long CoT	31.08(-61.29%)	72.86(-2.67%)	18.75(-10.54%)	6.67(0.00%)	32.34(-29.23%)
	MORALE	78.17(-2.64%)	73.30(-2.08%)	23.62(+12.69%)	10.00(+49.93%)	46.27(+1.26%)
Qwen2.5-7B	Long CoT	78.47(+6.27%)	82.54(+1.03%)	20.96(-28.73%)	20.00(+19.98%)	50.49(+0.17%)
	MORALE	92.27(+24.96%)	82.80(+1.35%)	29.78(+1.26%)	16.67(0.00%)	55.38(+9.87%)
Deepseek-7B	Long CoT	79.98(-0.39%)	46.92(+2.58%)	12.87(-16.65%)	3.33(-50.07%)	35.78(-3.40%)
	MORALE	80.70(+0.51%)	47.42(+3.67%)	15.81(+2.40%)	10.00(+49.93%)	38.48(+3.91%)
OPENR1-5K						
Qwen2.5-1.5B	Long CoT	58.07(-27.67%)	75.46(+0.80%)	9.56(-54.39%)	13.33(+99.85%)	39.11(-14.42%)
	MORALE	73.92(-7.93%)	76.23(+1.83%)	17.34(-17.27%)	20.00(+199.85%)	46.87(+2.58%)
Qwen2.5-7B	Long CoT	77.71(+5.24%)	82.78(+1.32%)	26.84(-8.74%)	16.67(0.00%)	51.00(+1.18%)
	MORALE	91.65(+24.12%)	84.42(+3.33%)	33.82(+14.99%)	16.67(0.00%)	56.64(+12.37%)
Deepseek-7B	Long CoT	83.78(+4.35%)	45.84(+0.22%)	16.44(+6.48%)	6.67(0.00%)	38.18(+3.10%)
	MORALE	83.24(+3.67%)	47.16(+3.10%)	19.19(+24.29%)	10.00(+49.93%)	39.90(+7.73%)

determine whether the sample was answered correctly. The evaluation framework we use is OpenCompass(Contributors, 2023), and we evaluate the model's single-inference accuracy, i.e., the *pass*@1 metric.

4.2 Overall Performance

Table 1 shows the overall performance of MORALE across different model families, model sizes, and dataset sizes. Our method consistently outperforms the conventional distillation method in all settings. We highlight several key observations below.

Small Models Struggle to Reasoning Consistent with the scaling law(Kaplan et al., 2020), models with more parameters consistently outperform those with fewer parameters when distilling long CoT across various datasets and benchmarks. Specifically, after distillation on the S1K dataset, Qwen-7B scored an average of 50.49, whereas Qwen-1.5B scored 32.34. On GSM8K, for instance, Qwen-7B scored 78.47, significantly outperforming Qwen-1.5B at 31.08. Analysis of the outputs indicated that after distillation, the limited capacity of Qwen-1.5B led to persistent repetition, hindering its ability to derive the correct final answer from the generated steps. A similar trend is also observed on Deepseek-7B, potentially due to its weaker capabilities. Conversely, larger models demonstrated a better capacity to assimilate this knowledge. This suggests that applying long CoT distillation to small models typically yields only marginal performance improvements and may even degrade their original capabilities on certain benchmarks, thereby failing to meet the objective of substantially enhancing reasoning ability.

Smaller Models Benefit More Our method demonstrates greater improvements for smaller models compared to larger models, suggesting that MORALE is particularly effective in enhancing the reasoning abilities of smaller models and more efficiently leveraging the potential of long CoT. Specifically, for the Qwen-7B model, our method further enhances the reasoning ability already benefiting from long CoT, offering an additional performance gain. However, for Qwen-1.5B, long CoT alone significantly degrades the model's capability, with the average score decreasing by 29.23%

Table 2: Ablation study on the effectiveness of our summary segments on Qwen-1.5B and Qwen-7B. "-*summary only*" indicates replacing the original thinking part of each segment with just the segment's summary. "-*draft*" indicates placing the summary before the segment, serving as a draft for problem-solving. The best result is indicated in **bold**.

Models	Method	GSM8K	MATH	DROP
Qwen-1.5B	MORALE(-RS2DPO)	**76.12**	**75.32**	**39.61**
	-summary only	69.52	66.36	36.10
	-draft	74.00	75.12	38.68
Qwen-7B	MORALE(-RS2DPO)	**92.95**	**84.90**	**69.28**
	-summary only	89.99	81.30	47.81
	-draft	80.29	84.36	68.61

compared to the vanilla version. In contrast, our method applied to Qwen-1.5B not only improves the model's ability to solve challenging problems but also mitigates the negative impact, resulting in an overall net improvement of 1.26%. Additionally, on the OpenR1-5K dataset and the AIME24 benchmark, the score for Qwen-1.5B with long CoT was 13.33. Applying MORALE increased this score significantly to 20.00. In comparison, for Qwen-7B, the score with MORALE was more limited to 16.67. Furthermore, extending beyond comparisons within the Qwen family, Qwen-1.5B consistently exhibits greater improvements across various benchmarks than Deepseek-7B.

Broad Applicability and Robust Generalization Despite its simplicity, MORALE demonstrates substantial adaptability, broad applicability, and robust generalization capabilities. Crucially, it eliminates the need for ground truth in the training dataset, substantially expanding its range of applications relative to methods requiring such data. This capability translates into significant performance gains across diverse model families and data scales. Notably, MORALE exhibits strong generalization, particularly on challenging out-of-domain benchmarks. For instance, beyond established benchmarks like GSM8K, MATH, and AIME24, which may be susceptible to over-optimization, MORALE performs strongly on demanding reasoning tasks including BBH and DROP. The additional results can be found in Table 5 in Appendix due to page constraints. As detailed in Sec. 4.1, this strong performance on unseen or demanding evaluations is underpinned by a training set primarily sourced from public datasets, without specific optimizations tailored to these benchmarks.

Table 3: Ablation study on the effectiveness of our RS2DPO module on Qwen-1.5B and Qwen-7B. The best result is indicated in **bold**.

Models	Method	MATH	Minerva-Math	BBH
Qwen-1.5B	MORALE	**73.30**	**23.62**	**26.99**
	w/o RS2DPO	72.94	19.12	26.62
Qwen-7B	MORALE	**82.80**	**29.78**	**57.12**
	w/o RS2DPO	82.68	21.69	56.15

4.3 Analysis

4.3.1 Ablation study

To further analyze the effectiveness of our method, we conduct ablation studies to assess the contributions of each component within MORALE.

Effectiveness of summary segment Since the use of a powerful LLM for segmenting long CoT into sections and providing per-section summaries introduces the possibility that an additional teacher model contributed external knowledge, potentially leading to improved results. To demonstrate that summaries following individual problem-solving steps facilitate the small model's absorption of complex knowledge, we conducted an ablation study on the inclusion and position of these summaries, using models without RS2DPO stage. During the knowledge distillation period, we exclude the thinking part, relying solely on the summary generated by the external powerful model, which we denote as "-*summary only*". Furthermore, We placed the summary before each segment, treating it as a draft to investigate the importance of summary position, which we refer to as "-*draft*". We report the accuracy on several representative tasks in Table 2. It is evident that **applying summary alone instead of thinking process results in a significant performance drop**. We attribute this decline to the *reduced specific verification and backtracing steps*, as the model relies solely on general thoughts covered in the summaries. Besides, **rearranging the position of summary degrades performance**. This decline highlights the essential role of summary: by enabling summarization during problem-solving process rather than making draft beforehand, it significantly enhances mathematical reasoning capabilities.

Effectiveness of RS2DPO Following the incorporation of long CoT knowledge, we employ RS2DPO to align the model's preference further. The aim is to *reinforce correct and concise problem-solving trajectories within the model's search space*

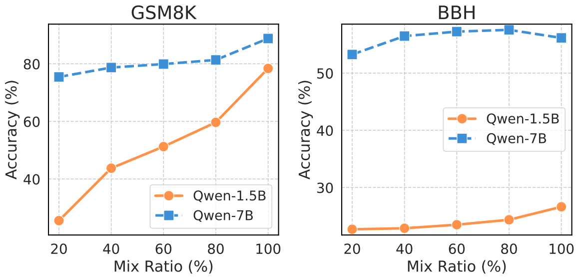


Figure 4: Performance vs. Mix Ratio. Higher summary segment ratios correlate with steady performance improvements, especially for Qwen-1.5B.

while suppressing erroneous, overly verbose, and unhelpful ones. To evaluate its effectiveness, we conducted an ablation study on the RS2DPO module, as shown in Table 3. The promising results demonstrate the effectiveness of RS2DPO to mitigate negative influences introduced by long CoT distillation, particularly in the context of small models.

4.3.2 Different ratio of summary segments

We also investigated the effect of varying the proportion (α) of data containing the summary segment when mixed with the original long CoT data. Specifically, the mixed dataset comprised a proportion α of examples containing the summary segment and a proportion of 1-α of original long CoT examples. The results are presented in Figure 4. Maintaining a constant total data volume, increasing the proportion of data containing the summary segment continues enhancing model performance on both GSM8K and BBH. Optimal performance was achieved when we included the summary segments fully ($\alpha = 1$), except for the Qwen-7B model on BBH. Specifically, it suggests that incorporating the summary segment is likely to maximize performance.

4.3.3 Relationship between token efficiency and performance

To investigate the mechanisms behind MORALE's effectiveness, we examined the relationship between output token length, average accuracy, and different methods, as depicted in Figure 5. Following long CoT distillation, the model's output approached 10,000 tokens. Incorporating summary segments during distillation further augmented output, likely necessitating continuous summarization during problem-solving. Concurrently, performance increased with output length, aligning with test-time scaling observations. In striking contrast,

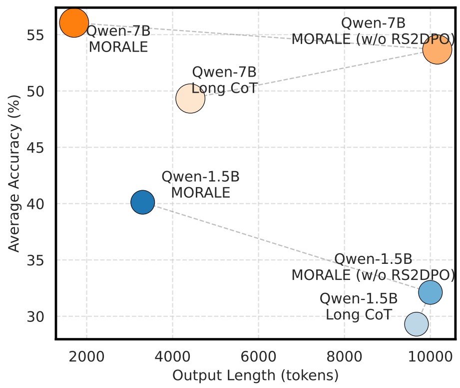


Figure 5: Relationship between average output length (tokens) and accuracy across different methods. Results show that MORALE achieves optimal performance while maintaining a low output token count.

the application of the RS2DPO module dramatically reduced output while concurrently enhancing performance. This discrepancy suggests that although small models develop improved reasoning capacity after learning from LRMs, uncontrolled iteration (e.g., verification, backtracking) can steer them towards inefficient or unproductive reasoning trajectories, impeding effective problem resolution. Fortunately, the proposed MORALE delivers efficient yet productive reasoning capacity.

5 Conclusion

This paper systematically investigates the negative impacts of long CoT distillation on small models. We propose the MORALE framework, which enables small models to better assimilate complex knowledge from long CoT data and achieve higher performance with fewer output tokens. Our extensive experimental analysis reveals that incorporating summary segments is consistent with scaling laws and is more effective in stimulating reasoning abilities. And the incorporation of RS2DPO module further enhances model performance while keeping output short. Our method achieves nearly consistent improvements across datasets, model scales, and benchmarks. These findings offer guidance for incorporating new datasets in future research and reduce practical deployment costs, thus demonstrating significant practical value.

Limitations

While this study provides valuable insights into the optimization of long CoT distillation, several limitations persist. Firstly, its focus has been solely on mathematical reasoning tasks, leaving its effectiveness and adaptability in other domains unexplored. Moreover, our long CoT data was exclusively sourced from R1, and other powerful LRMs such as QWQ(Team, 2025) were not utilized. This limited scope raises the question of whether our distillation method's performance scales with advancements in the underlying reasoning techniques. Exploring diverse sources of long CoT data to enhance the performance of MORALE represents a promising avenue for future work.

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*.

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

AI-MO. 2024. AIME 2024 dataset (validation set). Version 2024a.

David D Baek and Max Tegmark. 2025. Towards understanding distilled reasoning models: A representational approach. *arXiv preprint arXiv:2503.03730*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Qiguang Chen, Libo Qin, Jiaqi Wang, Jingxuan Zhou, and Wanxiang Che. 2024a. Unlocking the capabilities of thought: A reasoning boundary framework to quantify and optimize chain-of-thought. *Advances in Neural Information Processing Systems*, 37:54872–54904.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, and 1 others. 2024b. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, 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*.

OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. `https://github.com/open-compass/opencompass`.

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, and 1 others. 2025. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378.

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

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. 2023. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems*, 36:70757–70798.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. In *International Conference on Machine Learning*, pages 10421–10430. PMLR.

Zeyu Gan, Yun Liao, and Yong Liu. 2025. Rethinking external slow-thinking: From snowball errors to probability of correct reasoning. *arXiv preprint arXiv:2501.15602*.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. 2025. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*.

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*.

Yancheng He, Shilong Li, Jiaheng Liu, Weixun Wang, Xingyuan Bu, Ge Zhang, Zhongyuan Peng, Zhaoxiang Zhang, Zhicheng Zheng, Wenbo Su, and 1 others. 2025. Can large language models detect errors in long chain-of-thought reasoning? *arXiv preprint arXiv:2502.19361*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

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

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*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Gyeongman Kim, Doohyuk Jang, and Eunho Yang. 2024. Promptkd: Distilling student-friendly knowledge for generative language models via prompt tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6266–6282.

Abhinav Kumar, Jaechul Roh, Ali Naseh, Marzena Karpinska, Mohit Iyyer, Amir Houmansadr, and Eugene Bagdasarian. 2025. Overthink: Slowdown attacks on reasoning llms. *arXiv e-prints*, pages arXiv–2502.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

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).

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

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Jiaheng Liu, Dawei Zhu, Zhiqi Bai, Yancheng He, Huanxuan Liao, Haoran Que, Zekun Wang, Chenchen Zhang, Ge Zhang, Jiebin Zhang, and 1 others. 2025. A comprehensive survey on long context language modeling. *arXiv preprint arXiv:2503.17407*.

Yijia Luo, Yulin Song, Xingyao Zhang, Jiaheng Liu, Weixun Wang, GengRu Chen, Wenbo Su, and Bo Zheng. 2025. Deconstructing long chain-of-thought: A structured reasoning optimization framework for long cot distillation. *arXiv preprint arXiv:2503.16385*.

Yiran Ma, Zui Chen, Tianqiao Liu, Mi Tian, Zhuo Liu, Zitao Liu, and Weiqi Luo. 2025. What are step-level reward models rewarding? counterintuitive findings from mcts-boosted mathematical reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24812–24820.

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*.

Jiabao Pan, Yan Zhang, Chen Zhang, Zuozhu Liu, Hongwei Wang, and Haizhou Li. 2024. Dynathink: Fast or slow? a dynamic decision-making framework for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14686–14695.

Shubham Parashar, Blake Olson, Sambhav Khurana, Eric Li, Hongyi Ling, James Caverlee, and Shuiwang Ji. 2025. Inference-time computations for llm reasoning and planning: A benchmark and insights. *arXiv preprint arXiv:2502.12521*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741.

Leonardo Ranaldi and Andre Freitas. 2024. Aligning large and small language models via chain-of-thought reasoning. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1812–1827.

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

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2022. Distilling reasoning capabilities into smaller language models. *arXiv preprint arXiv:2212.00193*.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2023. Distilling reasoning capabilities into smaller language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7059–7073.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Hanjie Chen, Xia Hu, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and 1 others. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, and 1 others. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

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

Qwen Team. 2025. Qwq-32b: Embracing the power of reinforcement learning.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Chengyu Wang, Taolin Zhang, Richang Hong, and Jun Huang. 2025a. A short survey on small reasoning models: Training, inference, applications and research directions. *arXiv preprint arXiv:2504.09100*.

Yu Wang, Nan Yang, Liang Wang, and Furu Wei. 2025b. Examining false positives under inference scaling for mathematical reasoning. *arXiv preprint arXiv:2502.06217*.

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, and 1 others. 2025c. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yuchen Yan, Jin Jiang, Yang Liu, Yixin Cao, Xin Xu, Mengdi Zhang, Xunliang Cai, and Jian Shao. 2025. S^ 3cmath: Spontaneous step-level self-correction makes large language models better mathematical reasoners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25588–25596.

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

Wang Yang, Xiang Yue, Vipin Chaudhary, and Xiaotian Han. 2025. Speculative thinking: Enhancing small-model reasoning with large model guidance at inference time. *arXiv preprint arXiv:2504.12329*.

Zhe Yang, Yichang Zhang, Yudong Wang, Ziyao Xu, Junyang Lin, and Zhifang Sui. 2024b. Confidence vs critique: A decomposition of self-correction capability for llms. *arXiv preprint arXiv:2412.19513*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Huifeng Yin, Yu Zhao, Minghao Wu, Xuanfan Ni, Bo Zeng, Hao Wang, Tianqi Shi, Liangying Shao, Chenyang Lyu, Longyue Wang, and 1 others. 2025. Towards widening the distillation bottleneck for reasoning models. *arXiv preprint arXiv:2503.01461*.

Yong Zhang, Bingyuan Zhang, Zhitao Li, Ming Li, Ning Cheng, Minchuan Chen, Tao Wei, Jun Ma, Shaojun Wang, and Jing Xiao. 2025. Self-enhanced reasoning training: Activating latent reasoning in small models for enhanced reasoning distillation. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2024. Processbench: Identifying process errors in mathematical reasoning. *arXiv preprint arXiv:2412.06559*.

A More Experiments Settings

A.1 Segmentation Model

We selected the `Qwen2.5-32B-Instruct` model as the powerful language model for segmenting. During the training process, the maximum learning rate was set to 1.0e-5, the optimizer was chosen as AdamW, a cosine scheduler was adopted with a warm-up ratio of 0.1. Training utilized 16 Nvidia A100 GPUs, with a global batch size of 16, for 3 epochs. The maximum training length was 32768, totaling 117 steps, and the training duration was approximately 1 hour. The segmentation prompt we used is displayed in Figure 8.

A.2 SFT Stage

We set the initial learning rate to 1e-5, with a 0.05 warm-up ratio, and decay to 0 using a cosine scheduler. Following (Muennighoff et al., 2025), all models are trained for 5 epochs with a sequence length of 32768 tokens and a global batch size of 16. We conducted our SFT experiments on 16 Nvidia A100 GPUs.

A.3 RS2DPO Stage

In the RS2DPO part, we randomly sample 1K data from (LI et al., 2024) for RS. In the first stage, 4 trajectories were sampled, from which a trajectory with a correct response was selected as the input for the second stage. In the second stage, the number of final truncated segments k was set to 5 and 8 trajectories were sampled, from which the correct and short one was selected as the chosen, the wrong and lengthy one was selected as the rejected. We also experimented with varying sample attempt counts and the value of k, but the outcomes were comparable. Increasing the number of attempts leads to additional overhead, whereas decreasing it results in performance degradation. The selection of 4 attempts for the first stage and 8 attempts for the second stage was aimed at achieving a trade-off between performance and overhead. The DPO training was conducted for 2 epochs, with a learning rate of 5e-6 and incorporating an SFT loss weighted at 0.1.

B More Experiments Results

In this section, we additionally present the results for the Bespoke-Stratos-17K dataset.

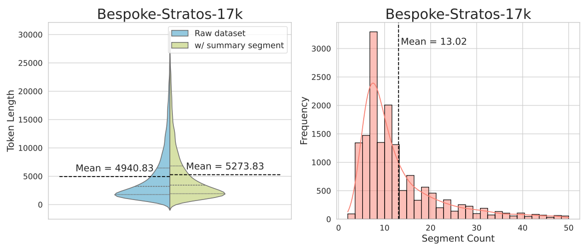


Figure 6: Detailed analysis of the Bespoke-Stratos-17K dataset we used. In the left, "Raw dataset" denotes the original open-source dataset, whereas "w/ summary segment" refers to the dataset after segmentation and the inclusion of summary segments. In the right, the number of segments after segmentation is showed.

B.1 Dataset Detail

Figure 6 shows the distributions of original data length and segmented data length for the Bespoke-Stratos-17K dataset. Furthermore, it also systematically displays the distribution of the number of segments per data entry. Specifically, after segment splitting, the data distribution trend remains largely unchanged, while the average length increases, indicating that longer thought processes are split into more independent segments, which is also consistent with intuition.

B.2 Additional Results

Table 4 presents the benchmark results for various models on the Bespoke-Stratos-17K dataset. As can be seen, the results are consistent with previous findings: relatively small and less capable models show only marginal improvements in capability after long CoT distillation, sometimes even decreasing. Upon using our method, the negative effects can be significantly mitigated, enhancing performance on challenging benchmarks.

C Data Examples

This section demonstrates several examples of segmentation inputs and outputs, raw long CoT data and data with segment summary.

C.1 Segmentation Example

We present the prompt used for segmentation (Figure 8) and an example including input and output below (Figure 9).

C.2 Long CoT Example

We present the complete result demonstrates in Figure 1, including the question, raw long CoT output (Figure 10) and our output with summary (Figure 11).

Table 4: More experiment results on Bespoke-Stratos-17K

Models	Method	*In-Domain*				*Out-Of-Domain*		Avg.
		GSM8K	**MATH**	**Minerva-Math**	**AIME24**	**BBH**	**DROP**	
Qwen2.5-1.5B	Vanilla	80.29	74.86	20.96	6.67	16.81	39.78	39.90
Qwen2.5-7B	Vanilla	73.84	81.70	29.41	16.67	39.40	67.17	51.37
Deepseek-7B	Vanilla	80.29	45.74	15.44	6.67	48.57	71.10	44.64
BESPOKE-STRATOS-17K								
Qwen2.5-1.5B	Long CoT	39.27 (-51.09%)	62.72 (-16.22%)	13.60 (-35.11%)	3.33 (-50.07%)	17.79 (+5.83%)	39.09 (-1.74%)	29.30 (-26.57%)
	MORALE	71.28 (-11.22%)	75.46 (+0.80%)	23.16 (+10.50%)	13.33 (+99.85%)	18.48 (+9.94%)	39.01 (-1.94%)	40.12 (+0.55%)
Qwen2.5-7B	Long CoT	71.65 (-2.96%)	83.18 (+1.81%)	26.84 (-8.74%)	13.33 (-20.03%)	30.97 (-21.39%)	70.02 (+4.24%)	49.33 (-3.97%)
	MORALE	93.03 (+26.00%)	84.00 (+2.82%)	36.40 (+23.77%)	23.33 (+40.01%)	29.94 (-24.01%)	69.66 (+3.70%)	56.06 (+9.13%)
Deepseek-7B	Long CoT	84.61 (+5.40%)	52.20 (+14.12%)	23.90 (+54.79%)	6.67 (0.00%)	24.73 (-49.08%)	68.48 (-3.70%)	43.43 (-2.71%)
	MORALE	82.41 (+2.64%)	56.08 (+22.61%)	24.63 (+59.52%)	6.67 (0.00%)	25.75 (-46.98%)	72.73 (+2.30%)	44.71 (+0.16%)

Table 5: More experiment results on out-of-domain benchmarks

Models	Method	BBH	DROP	Avg.
Qwen2.5-1.5B	Vanilla	16.81	39.78	28.30
Qwen2.5-7B	Vanilla	39.4	67.17	53.29
Deepseek-7B	Vanilla	48.57	71.1	59.84
S1K				
Qwen2.5-1.5B	Long CoT	22.85 (+35.93%)	36.15 (-9.13%)	29.50 (+4.26%)
	MORALE	26.99 (+60.56%)	36.05 (-9.38%)	31.52 (+11.40%)
Qwen2.5-7B	Long CoT	51.25 (+30.08%)	67.37 (+0.30%)	59.31 (+11.31%)
	MORALE	57.12 (+44.97%)	66.52 (-0.97%)	61.82 (+16.02%)
Deepseek-7B	Long CoT	43.15 (-11.16%)	57.85 (-18.64%)	50.50 (-15.60%)
	MORALE	50.88 (+4.76%)	67.27 (-5.39%)	59.08 (-1.27%)
OPENR1-5K				
Qwen2.5-1.5B	Long CoT	18.48 (+9.93%)	39.01 (-1.94%)	28.75 (+1.59%)
	MORALE	18.97 (+12.85%)	37.49 (-5.76%)	28.23 (-0.23%)
Qwen2.5-7B	Long CoT	38.81 (-1.50%)	67.83 (+0.98%)	53.32 (+0.07%)
	MORALE	45.19 (+14.70%)	70.05 (+4.29%)	57.62 (+8.14%)
Deepseek-7B	Long CoT	27.14 (-44.12%)	62.20 (-12.52%)	44.67 (-25.34%)
	MORALE	39.05 (-19.60%)	71.70 (+0.84%)	55.38 (-7.45%)

Bad Case

[QUESTION]
Consider the sequence of numbers: $4, 7, 1, 8, 9, 7, 6, \ldots$ For $n > 2$, the n-th term of the sequence is the units digit of the sum of the two previous terms. Let S_n denote the sum of the first n terms of this sequence. Find the smallest value of n for which $S_n > 10,000$.

[ANSWER]
<think>
Okay, so I need to solve this problem where there's a sequence starting with 4, 7, and then each subsequent term is the units digit of the sum of the two previous terms. We have to find the smallest n where the sum of the first n terms exceeds 10,000. Hmm, let's break this down step by step.
First, let me make sure I understand the sequence correctly. The first two terms are given as 4 and 7. Then, for n > 2, each term is the units digit of the sum of the two previous terms. So, the third term would be the units digit of (4 + 7), which is 11, so the units digit is 1. Then the fourth term is the units digit of (7 + 1) = 8. The fifth term is the units digit of (1 + 8) = 9, and so on. Let me write out the first few terms to see the pattern:

1. 4
2. 7
3. 4 + 7 = 11 $\rightarrow$ units digit 1
4. 7 + 1 = 8 $\rightarrow$ units digit 8
5. 1 + 8 = 9 $\rightarrow$ units digit 9
6. 8 + 9 = 17 $\rightarrow$ units digit 7
7. 9 + 7 = 16 $\rightarrow$ units digit 6
8. 7 + 6 = 13 $\rightarrow$ units digit 3
9. 6 + 3 = 9 $\rightarrow$ units digit 9
10. 3 + 9 = 12 $\rightarrow$ units digit 2
11. 9 + 2 = 11 $\rightarrow$ units digit 1
12. 2 + 1 = 3 $\rightarrow$ units digit 3
13. 1 + 3 = 4 $\rightarrow$ units digit 4
14. 3 + 4 = 7 $\rightarrow$ units digit 7
15. 4 + 7 = 11 $\rightarrow$ units digit 1
16. 7 + 1 = 8 $\rightarrow$ units digit 8
17. 1 + 8 = 9 $\rightarrow$ units digit 9
18. 8 + 9 = 17 $\rightarrow$ units digit 7
19. 9 + 7 = 16 $\rightarrow$ units digit 6
20. 7 + 6 = 13 $\rightarrow$ units digit 3
21. 6 + 3 = 9 $\rightarrow$ units digit 9
22. 3 + 9 = 12 $\rightarrow$ units digit 2
23. 9 + 2 = 11 $\rightarrow$ units digit 1
24. 2 + 1 = 3 $\rightarrow$ units digit 3
25. 1 + 3 = 4 $\rightarrow$ units digit 4

...

Figure 7: An bad case of repetitive outputs and logical confusion

Segmentation Prompt

You are an intelligent assistant. You need to read the problem-solving process below, which is divided into multiple steps using line breaks, with each step labeled by a tag. Then, combine them into multiple logically independent sections. Specific requirements:
1. Each section should be a complete atomic problem-solving process or an attempt.
2. When aggregating sections, you need to output the start and end step tags (they can be the same) and provide a brief summary or title.
3. Please output in the following format: «start tag» - «end tag» Brief description.
The end tag should be greater than or equal to the start tag and the start tag of a section should be the end tag of the previous section plus one.
Below is the problem-solving process:
[PROCESS]
question
[/PROCESS]
Please give me your output:

Figure 8: Segmentation prompt for data curation

Segmentation Input

You are an intelligent assistant. You need to read the problem-solving process below, which is divided into multiple steps using line breaks, with each step labeled by a tag. Then, combine them into multiple logically independent sections.
Specific requirements:
1. Each section should be a complete atomic problem-solving process or an attempt.
2. When aggregating sections, you need to output the start and end step tags (they can be the same) and provide a brief summary or title.
3. Please output in the following format: «start tag» - «end tag» Brief description.
The end tag should be greater than or equal to the start tag and the start tag of a section should be the end tag of the previous section plus one.
Below is the problem-solving process:
[PROCESS]
«1» Okay, let's see. So there's this problem about a store buying pens and selling them at different prices, but the profit is the same in both scenarios. The question is asking for the purchase price per pen. Hmm. Let me try to break this down step by step.
«2» First, let's understand the problem. The store bought a bunch of pens. Then they sold 20 pens at 7 yuan each and another 15 pens at 8 yuan each. In both cases, the profit earned is the same. So, selling 20 pens at 7 yuan gives the same profit as selling 15 pens at 8 yuan. We need to find out how much each pen cost the store to purchase.
«3» Alright, so profit is calculated as (selling price - cost price) multiplied by the number of pens sold. Since both scenarios result in the same profit, I can set up two equations and then equate them. Let me write that out.
«4» Let's denote the cost price per pen as C yuan. That's our unknown.
«5» In the first case, selling 20 pens at 7 yuan each. So the revenue here is 20 * 7. The cost for those 20 pens would be 20 * C. Therefore, profit is revenue minus cost, which is 20*7 - 20*C.
«6» In the second case, selling 15 pens at 8 yuan each. Similarly, revenue is 15*8, and cost is 15*C. So profit is 15*8 - 15*C.
«7» According to the problem, these two profits are equal. So:
«8» 20*7 - 20*C = 15*8 - 15*C
«9» Now, let me compute the numerical values for the revenues first.
«10» First scenario: 20 pens at 7 yuan is 140 yuan. Second scenario: 15 pens at 8 yuan is 120 yuan. So the equations become:
«11» 140 - 20C = 120 - 15C
«12» Now, I need to solve for C. Let's rearrange the equation. Let's get all the terms with C on one side and constants on the other.
«13» 140 - 120 = 20C - 15C
«14» 20 = 5C
«15» Divide both sides by 5:
«16» C = 4
«17» Wait, so the cost price per pen is 4 yuan? Let me check that again to make sure I didn't make a mistake.

Segmentation Input

«18» Plugging back into the original equations. First profit: 20*(7 - 4) = 20*3 = 60 yuan. Second profit: 15*(8 - 4) = 15*4 = 60 yuan. Yes, both profits are 60 yuan. That checks out. So the purchase price per pen is 4 yuan.
«19» But let me just go through the algebra again to be thorough.
«20» Starting equation:
«21» 20*7 - 20C = 15*8 - 15C
«22» 140 - 20C = 120 - 15C
«23» Subtract 120 from both sides:
«24» 20 - 20C = -15C
«25» Add 20C to both sides:
«26» 20 = 5C
«27» So C = 4. Yep, that's correct.
«28» Alternatively, maybe think about the difference in the number of pens and the prices. Let's see. When they sell 20 pens at 7 yuan, versus 15 pens at 8 yuan. The difference in revenue is 140 - 120 = 20 yuan. But the profits are the same. So that extra 20 yuan in revenue must be offset by the extra cost of the pens sold. Since they sold 5 more pens in the first scenario (20 vs 15), the extra cost would be 5C. Therefore, 20 yuan extra revenue minus 5C extra cost equals the same profit. So:
«29» 20 - 5C = 0
«30» Which gives 5C = 20 => C = 4. Same result. That's another way to think about it.
«31» So either way, the answer is 4 yuan per pen. I think that makes sense. Let me just verify once more with substitution.
«32» If each pen cost 4 yuan, then:
«33» First scenario: sells 20 pens at 7 yuan. Revenue is 20*7 = 140. Cost is 20*4 = 80. Profit is 140 - 80 = 60.
«34» Second scenario: sells 15 pens at 8 yuan. Revenue is 15*8 = 120. Cost is 15*4 = 60. Profit is 120 - 60 = 60.
«35» Yep, both profits are 60. So it all checks out. Therefore, the purchase price per pen is indeed 4 yuan.
«36» **Final Answer**
The purchase price of each pen is $\boxed{4}$ yuan.
[/PROCESS]
Please give me your output:

Segmentation Output

«1» - «2» Introduction and understanding the problem
«3» - «8» Setting up the profit equations for both scenarios
«9» - «11» Calculating revenue and forming simplified equations
«12» - «16» Solving the simplified equation for cost price per pen (C)
«17» - «18» Verification of solution by plugging back into original equations
«19» - «27» Re-solving the equation step-by-step for thoroughness
«28» - «30» Alternative reasoning method to confirm the result
«31» - «35» Final verification by substituting the cost price back into both scenarios
«36» - «36» Conclusion and final answer: purchase price per pen is 4 yuan

Figure 9: A example of output after segmentation

Question

What is the product of $0.\overline{05}$ and 1.8 expressed as a fraction in simplest form?

Raw Long CoT Output

<think>
Okay, so I need to find the product of 0.05 repeating and 1.8, and then express that as a fraction in simplest form. Hmm, let me start by figuring out how to convert 0.05 repeating into a fraction. I remember that there's a method for converting repeating decimals into fractions, but I might have to write that out step by step to make sure I do it correctly.

First, let me denote 0.05 repeating as x. So, x = 0.05555... Now, to eliminate the repeating part, I can multiply x by 100 to shift the decimal point two places to the right. That would make 100x = 5.55555... Then, if I subtract the original x from this equation, I can eliminate the repeating part. Let me write that out:

100x = 5.55555...

- x = 0.05555...

= 5.55555... - 0.05555... = 5.5

So, 100x - x = 5.5, which simplifies to 99x = 5.5. Then, solving for x, I get x = 5.5 / 99. Hmm, 5.5 divided by 99. To make this a fraction, I can multiply numerator and denominator by 10 to eliminate the decimal in the numerator: 55/990. Then, I can simplify that fraction by dividing numerator and denominator by their greatest common divisor. Let me check the GCD of 55 and 990. Both are divisible by 5, so 55 ÷ 5 = 11, and 990 ÷ 5 = 198. So now, it's 11/198. Is 11 and 198 coprime? Let's see, 198 divided by 11 is 18, so yes, 11 is prime and doesn't divide 198. Therefore, 11/198 is the simplified fraction for 0.05 repeating. Wait, but 11/198 can be simplified further? Let me check again. 198 divided by 2 is 99, 99 divided by 3 is 33, 33 divided by 3 is 11. So prime factors of 198 are 2, 3, 3, 11. The numerator is 11, which is prime. So 11 and 198 share no common factors other than 1. So 11/198 is indeed the simplest form. Okay, so 0.05 repeating is 11/198.

Now, the next part is multiplying that by 1.8. Let me convert 1.8 into a fraction. 1.8 is the same as 1 and 8/10, which simplifies to 1 and 4/5, or 9/5 when converted to an improper fraction. Let me confirm: 1.8 = 1 + 0.8 = 1 + 4/5 = 5/5 + 4/5 = 9/5. Yes, that's correct.

So now, I need to multiply 11/198 by 9/5. Let's do that. Multiplying numerators: 11 * 9 = 99. Multiplying denominators: 198 * 5 = 990. So the product is 99/990. Now, I need to simplify this fraction. Let's find the GCD of 99 and 990. Both are divisible by 99. 99 ÷ 99 = 1, and 990 ÷ 99 = 10. So 99/990 simplifies to 1/10. Wait, is that right? 99 divided by 99 is 1, and 990 divided by 99 is 10, so yes, 1/10. Let me check again. 99 and 990: 990 ÷ 99 = 10. So yes, GCD is 99. Therefore, 99/990 reduces to 1/10. So the product is 1/10.

Wait, but let me double-check all the steps to make sure I didn't make a mistake. First, converting 0.05 repeating to 11/198: that seems right. Then 1.8 is 9/5: correct. Multiplying 11/198 * 9/5 gives 99/990: correct. Then simplifying by dividing numerator and denominator by 99 gives 1/10: that's correct. So the final answer should be 1/10.

But wait, let me verify by another method. Maybe convert 0.05 repeating to a decimal and multiply by 1.8. Let's see, 0.05555... multiplied by 1.8. Let's compute that.

1.8 * 0.05555... Let's break it down. 1.8 * 0.05 = 0.09, and 1.8 * 0.005555... = 0.009999... So adding those together: 0.09 + 0.009999... = 0.099999..., which is 0.1. Because 0.099999... is equal to 0.1. So 0.05555... * 1.8 = 0.1. Which is 1/10. So that confirms the answer. So that's a good check. So 1/10 is correct.

Raw Long CoT Output

Therefore, the product of 0.05 repeating and 1.8 expressed as a fraction in simplest form is 1/10.
Final Answer
The product is $\boxed{\frac{1}{10}}$.
</think>
To find the product of $0.\overline{05}$ and 1.8 expressed as a fraction in simplest form, we start by converting $0.\overline{05}$ into a fraction.
Let $x = 0.\overline{05}$. Multiplying both sides by 100 to shift the decimal point two places to the right, we get:

$$100x = 5.\overline{05}$$

Subtracting the original x from this equation:

$$100x - x = 5.\overline{05} - 0.\overline{05}$$

$$99x = 5$$

Solving for x:

$$x = \frac{5}{99}$$

To simplify $\frac{5}{99}$, we convert $0.\overline{05}$ to a fraction:

$$0.\overline{05} = \frac{5}{99}$$

Next, we convert 1.8 to a fraction:

$$1.8 = \frac{9}{5}$$

Multiplying the two fractions:

$$\frac{5}{99} \times \frac{9}{5} = \frac{5 \times 9}{99 \times 5} = \frac{45}{495}$$

Simplifying $\frac{45}{495}$ by dividing the numerator and denominator by their greatest common divisor, which is 45:

$$\frac{45 \div 45}{495 \div 45} = \frac{1}{11}$$

Wait, correction: Simplifying $\frac{45}{495}$ by dividing the numerator and denominator by their greatest common divisor, which is 45:

$$\frac{45 \div 45}{495 \div 45} = \frac{1}{11}$$

Raw Long CoT Output

Wait, that's incorrect. Wait, 495 divided by 45 is 11, and 45 divided by 45 is 1. So, $\frac{45}{495} = \frac{1}{11}$. Wait, but that contradicts the earlier check. Wait, but 45 divided by 45 is 1, and 495 divided by 45 is 11. So, $\frac{45}{495} = \frac{1}{11}$. Wait, but that can't be right because 0.05 repeating times 1.8 is 0.1, which is $\frac{1}{10}$. Wait, so where did I go wrong?

Wait, let's recompute the multiplication step. Let's do 5/99 times 9/5. 5 times 9 is 45, and 99 times 5 is 495. So 45/495. Then, simplifying 45/495: both numerator and denominator are divisible by 45. 45 ÷ 45 = 1, and 495 ÷ 45 = 11. So 45/495 simplifies to 1/11. But that contradicts the decimal multiplication result. Wait, but 0.05555... times 1.8 is 0.1, which is 1/10. So which is correct?

Wait, let's check the decimal multiplication again. 0.05555... multiplied by 1.8. Let's compute 0.05555... * 1.8:

1.8 * 0.05 = 0.09

1.8 * 0.005555... = 0.009999...

Adding these together: 0.09 + 0.009999... = 0.099999..., which is 0.1. So 0.1 is 1/10. Therefore, the product is 1/10. So where did I go wrong in the fraction multiplication?

Wait, 0.05 repeating is 5/99, and 1.8 is 9/5. Multiplying these gives (5/99) * (9/5) = (5*9)/(99*5) = 45/495. Then, simplifying 45/495: divide numerator and denominator by 45: 45 ÷ 45 = 1, 495 ÷ 45 = 11. So 1/11. But that contradicts the decimal result. Wait, so which is correct?

Wait, 5/99 * 9/5 is (5*9)/(99*5) = 45/495. Then, 45 divided by 45 is 1, and 495 divided by 45 is 11. So 1/11. But 0.05555... * 1.8 = 0.1, which is 1/10. So which is correct?

Wait, maybe I made a mistake in converting 0.05555... to a fraction. Let me check again. Let x = 0.05555... Then, 100x = 5.55555... Subtract x: 100x - x = 5.55555... - 0.05555... = 5.5. So 99x = 5.5, so x = 5.5/99 = 55/990 = 11/198. Wait, 5.5 is 11/2, so 11/2 divided by 99 is 11/(2*99) = 11/198. Then, 11/198 * 9/5 = (11*9)/(198*5) = 99/990. Then, 99 and 990 have a GCD of 99: 99 ÷ 99 = 1, 990 ÷ 99 = 10. So 99/990 simplifies to 1/10. So that's correct. So 1/10 is the answer. But why did the decimal multiplication give 1/10? Because 0.05555... is a repeating decimal, and when multiplied by 1.8, it results in 0.1, which is 1/10. So both methods confirm that the answer is 1/10. Therefore, the correct answer is 1/10.

Final Answer

The product is $\boxed{\dfrac{1}{10}}$.

Figure 10: An example of raw Long CoT output, the final answer is wrong

MORALE Output

<think>
Okay, so I need to find the product of 0.05 repeating and 1.8, and then express that as a fraction in simplest form. Hmm, let's start by figuring out how to convert 0.05 repeating into a fraction. I remember that repeating decimals can be converted to fractions using algebra, but I might have to recall the exact method.
Let me write down 0.05 repeating. The repeating part is '05', so it's 0.050505... right? So, let's call this number x. So, x = 0.050505...
To convert this to a fraction, I think the trick is to multiply x by a power of 10 that shifts the decimal point past the repeating part. Since the repeating part is two digits, multiplying by 100 should work. Let me try that.
Multiply both sides by 100: 100x = 5.050505...
Now, subtract the original x from this equation to eliminate the repeating part.
So, 100x - x = 5.050505... - 0.050505...
That simplifies to 99x = 5. So, x = 5/99. Wait, is that right? Let me check. If I divide 5 by 99, does it repeat as 0.050505...? Let's see, 5 divided by 99. 99 goes into 5.000... 0 times. So, write down 0. and then 50 divided by 99 is 0.505050..., so yes, that's correct. So, 0.05 repeating is 5/99.
<summary> Convert the repeating decimal 0.05 repeating to a fraction </summary>
Okay, so now I need to multiply that by 1.8. Let's write 1.8 as a fraction. 1.8 is the same as 1 and 8/10, which simplifies to 1 and 4/5, or 9/5. Wait, 8/10 reduces to 4/5, so 1 + 4/5 is 9/5. Yes, that's correct. So, 1.8 is 9/5.
<summary> Convert 1.8 to a fraction </summary>
So now, the product is (5/99) multiplied by (9/5). Let me write that out:
(5/99) * (9/5)
I can simplify this by canceling out common factors. The 5 in the numerator of the first fraction cancels with the 5 in the denominator of the second fraction. Then, the 9 in the numerator of the second fraction cancels with the 99 in the denominator of the first fraction. Wait, 99 is 9*11, so 9 cancels with the 9, leaving 11 in the denominator. So, after canceling, we have 1/11.
<summary> Multiply the fractions and simplify </summary>
Wait, let me check that again. 5 divided by 5 is 1, and 9 divided by 99. 99 divided by 9 is 11, so yes, 99/9 is 11. So, the simplified fraction is 1/11. Let me verify this by multiplying numerators and denominators:
5*9 = 45 and 99*5 = 495. So, 45/495. Then, divide numerator and denominator by 45: 45 ÷ 45 = 1, 495 ÷ 45 = 11. So, 1/11. Yep, that's correct.
<summary> Verify the simplification by multiplying and dividing </summary>
Therefore, the product of 0.05 repeating and 1.8 is 1/11 when expressed as a fraction in simplest form.
<summary> Conclude the product as 1/11 </summary>
</think>
To find the product of $0.\overline{05}$ and 1.8 as a fraction in simplest form:
1. Convert $0.\overline{05}$ to a fraction: Let $x = 0.\overline{05}$. Then $100x = 5.\overline{05}$. Subtracting gives $99x = 5$, so $x = \frac{5}{99}$.
2. Convert 1.8 to a fraction: $1.8 = \frac{9}{5}$.
Multiply the fractions:

$$\frac{5}{99} \times \frac{9}{5} = \frac{45}{495} = \frac{1}{11}$$

The product is $\boxed{\frac{1}{11}}$.

Figure 11: An example of MORALE output, the final answer is correct