Weak2Wise: An Automated, Lightweight Framework for Weak-LLM-Friendly Reasoning Synthesis

Anonymous ACL submission

Abstract

Recent advances in large language model (LLM) fine-tuning have shown that incorporating high-quality reasoning traces into training data can markedly improve downstream performance. However, existing approaches often depend on expensive manual annotations or auxiliary models, and fail to adapt to the unique limitations of smaller "weak" LLMs. To address these gaps, we introduce **Weak2Wise**, a fully automated, lightweight framework for synthesizing high-quality, weak-LLM-friendly reasoning traces. Starting from a QA dataset, Weak2Wise filters out the samples that can already be correctly answered by the weak LLM, gathers diverse candidate reasoning traces from multiple strong LLMs, and leverages our Step-Mask scoring to rank and truncate the most guidance-effective traces. These reasoning traces are then used for fine-tuning, yielding substantial improvements in the weak LLM's reasoning abilities. The name Weak2Wise has two meanings: using a "weak" LLM to select the "wisest" reasoning traces generated by stronger LLMs, and fine-tuning the same weak LLM on these reasoning traces to become "wiser". We further use Weak2Wise to build GR-1K, a 1,000-sample math and science QA-reasoning dataset optimized for weak LLMs, and fine-tune Qwen2.5-7B on it to create GR-7B, which achieves superior performance on AIME2024, MATH-500, and GPQA Diamond benchmarks. Source code, dataset, and pretrained models will be made publicly available upon acceptance.

1 Introduction

The quality of training data plays a critical role in the fine-tuning of large language models (LLMs). During the fine-tuning stage, injecting reasoning traces into the training data has been shown to effectively improve the reasoning capabilities of LLMs (Hsieh et al., 2023; Shridhar et al., 2023; Li et al., 2023a; Yue et al., 2024). A high-quality

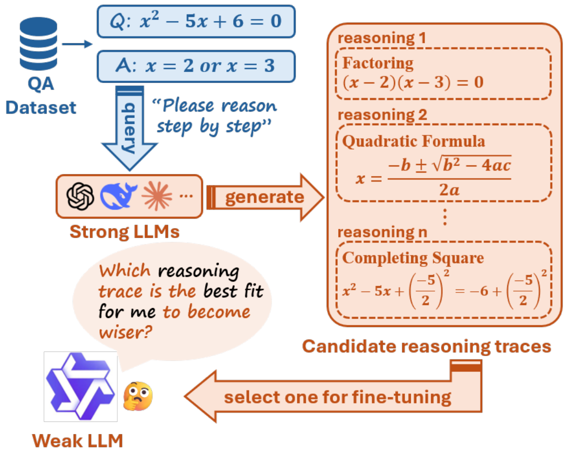

Figure 1: An example of applying Weak2Wise to synthesize reasoning for weak LLMs. Given a question and its answer, different strong LLMs generate various candidate reasoning traces. The key issue is to identify the reasoning trace that is truly suitable for fine-tuning a weak LLM. Note that the mathematical question in the figure is merely used to illustrate the pipeline, while actual questions we used are much more difficult.

dataset can significantly enhance the performance of an LLM in a specific domain, even with a limited number of examples (Zhou et al., 2023; Muennighoff et al., 2025).

To obtain high-quality reasoning traces for fine-tuning, researchers have explored automated methods beyond costly manual annotations. Some methods (Shao et al., 2023; Liu et al., 2023; Zelikman et al., 2024) automate the majority of the pipeline, yet still rely on manual intervention at critical steps .Other methods (Bhan et al., 2024; Lupidi et al., 2024; Haji et al., 2024) achieve full end-to-end automation but require training additional models, thereby compromising lightweight design.

However, weak LLMs[1] often follow different reasoning patterns from strong LLMs', making

[1]In this paper, we refer to LLMs with fewer than 10B parameters and without reasoning mode as weak LLMs.

them unable to interpret certain traces (Zhang et al., 2024; Hu et al., 2024; Li et al., 2024). Only few methods (Cai et al., 2025; Kim et al., 2025) note this gap and adjust their synthesis processes to fit reasoning patterns of weak LLMs. But they merely replicate these patterns rather than truly integrating weak LLMs into the synthesis pipeline.

This raises a natural and important question: *How can we synthesize high-quality reasoning traces with an automated and lightweight method, while ensuring that these reasoning traces are truly friendly to weak LLMs?*

To address this challenge, we propose **Weak2Wise**, an automated and lightweight framework for generating high-quality reasoning traces friendly to weak LLMs. The key issue is to identify the reasoning trace that is truly suitable for fine-tuning weak LLMs, as illustrated in Figure 1. Unlike prior methods that rely on external reward models or multi-agent coordination, Weak2Wise leverages the weak LLM's own performance to evaluate reasoning traces. The core idea is to evaluate each candidate reasoning trace's guidance effectiveness using a novel Step-Mask Scoring: incrementally masking partial reasoning steps and querying the weak LLM itself reveals how well each trace aids the weak LLM in reaching the correct answer.

Weak2Wise offers several distinct advantages: i) *Full Automation*: the entire pipeline is fully automated, requiring no human intervention; ii) *Lightweight Design*: no additional reward models or agent-based frameworks are required, keeping the entire process simple and scalable; and iii) *Weak-LLM-Friendly*: by leveraging the weak LLM's own behavior during evaluation, synthetic reasoning traces are adapted to reasoning patterns of weak LLMs.

We evaluate Weak2Wise on some of the most challenging mathematical and scientific reasoning benchmarks, including AIME2024 (Maxwell-Jia, 2024), MATH-500 (Lightman et al., 2023), and GPQA Diamond (Rein et al., 2023). Applied to the weak LLM Qwen2.5-7B (Yang et al., 2024), our framework achieves consistent and substantial improvements across all datasets. Ablation studies further verify the effectiveness of the Step-Mask Scoring and truncation strategies.

In summary, we make the following major contributions:

- We propose **Weak2Wise**, a new lightweight framework that fully automates the synthesis of reasoning traces friendly to weak LLMs. To the best of our knowledge, this is the first framework to incorporate the weak LLM itself into the reasoning synthesis process.

- In the Weak2Wise framework, we introduce a novel and pivotal method for evaluating different reasoning traces: **Step-Mask Scoring**. The resulting step-mask scores accurately reflect a weak LLM's comprehension of each trace, allowing us to identify the reasoning trace that best aligns with the weak LLM's reasoning patterns.

- We apply Weak2Wise to augment and construct a mathematics and science question-reasoning–answer dataset **GR-1K**, which contains 1,000 high-quality samples friendly to weak LLMs. We further fine-tune Qwen2.5-7B on GR-1K to obtain **GR-7B**, which achieves superior performance on reasoning-related evaluation tasks.

2 Related Work

2.1 Reasoning Augmentation via Strong LLM Prompting

A number of studies have demonstrated that it is effective for reasoning augmentation to distill prompting from strong LLMs to smaller models through fine-tuning. **Fine-tune-CoT** (Ho et al., 2023) prompts GPT-3.5 to produce multiple high-quality reasoning traces per QA pair and fine-tunes a student model on the resulting triples, yielding substantial reasoning gains. **SCoTD** (Li et al., 2023b) extends this by sampling diverse CoT traces from a large teacher and supervising the student on all variants, significantly boosting both supervised and few-shot performance. **SCOTT** (Wang et al., 2023) further introduces a counterfactual consistency objective to ensure the student truly relies on the provided chains. **KARD** (Kang et al., 2023) augments teacher reasoning traces with retrieved evidence before distillation, achieving strong improvements on knowledge-intensive tasks. **PaD** (Zhu et al., 2024) replaces free-form CoT with structured, executable programs to reduce noise and improve supervision fidelity. The above studies confirm that using higher-quality reasoning traces during fine-tuning leads to greater improvements in the target model's reasoning capabilities.

Method	Fully-Automated	Lightweight	Weak-LLM-Friendly
Auto-CoT	✓	✓	✗
Synthetic Prompting	✗	✓	✗
LogiCoT	✗	✓	✗
Self-AMPLIFY	✓	✗	✗
STaR	✗	✓	✗
Source2Synth	✓	✗	✗
MA-ToT	✓	✗	✗
Weak2Wise (Ours)	✓	✓	✓

Table 1: Comparison of high-quality reasoning synthesis methods in Section 2.2 and ours.

2.2 Synthesis of High-Quality Reasoning

Research on automated synthesis of high-quality reasoning traces has led to several innovative frameworks. **Auto-CoT** (Zhang et al., 2022) clusters questions and uses a strong LLM to generate exemplar chains per cluster, matching human-crafted prompts without manual annotations. **Synthetic Prompting** (Shao et al., 2023) bootstraps QA-CoT pairs by alternating backward question generation and forward reasoning generation to create large synthetic datasets. **LogiCoT** (Liu et al., 2023) uses meta-instructions to GPT-4 to produce a logically structured CoT dataset for instruction tuning. **Self-AMPLIFY** (Bhan et al., 2024) extracts post-hoc mini reasoning traces from both successful and failed cases to serve as demonstrations. **STaR** (Zelikman et al., 2024) iteratively leverages a small number of reasoning examples and a large dataset without reasoning, to bootstrap the ability to perform successively more complex reasoning. **Source2Synth** (Lupidi et al., 2024) generates synthetic data points with intermediate reasoning steps grounded in real-world sources and improves dataset quality by discarding low-quality generations based on their answer ability. **MA-ToT** (Haji et al., 2024) combines multi-agent reasoning with Tree-of-Thoughts (Yao et al., 2023) and introduces a Thought Verifier agent to filter out flawed reasoning branches.

Although these methods eliminate manual crafting, they often require clustering or meta-instruction design (not fully automated), train additional models (not lightweight), or synthesize reasoning that does not specially designed for the fine-tuned student model (lacking model specificity). Table 1 demonstrates the superiority of our **Weak2Wise** method compared with existing methods.

2.3 Weak-LLM-Friendly Reasoning

Various studies (Zhang et al., 2024; Hu et al., 2024; Li et al., 2024) have shown that weak LLMs exhibit different reasoning patterns from strong LLMs, motivating the need for synthesizing reasoning traces friendly to weak LLMs. Few recent methods (Cai et al., 2025; Kim et al., 2025) generate reasoning tailored to weak LLMs. However, they do not incorporate the weak LLM's actual performance into the reasoning optimization process. As a result, these approaches cannot obtain authentic feedback from the weak LLM, nor can they truly synthesize weak-LLM-Friendly reasoning traces. Our approach addresses this gap by integrating the weak LLM's real-time reasoning performances into an automated, lightweight synthesis framework, ensuring that the final reasoning traces are both high-quality and truly friendly to the weak LLM.

3 Method

3.1 Overview

Let $\mathcal{S}_{QA}$ denote an existing question–answer dataset, M_{weak} a base LLM with weak reasoning ability for selection and subsequent fine-tuning, and $\mathcal{M}_{\text{strong}}$ a set of strong reasoning LLMs used to generate candidate reasoning traces. Our method consists of five successive stages: (i) *Question-Answer Data Filtering*. Each $(q, a) \in \mathcal{S}_{QA}$ is evaluated by M_{weak}. Retain only those pairs for which M_{weak} produces an incorrect response, resulting in the filtered subset $\mathcal{S}'_{QA}$ (Section 3.2). (ii) *Candidate Reasoning Traces Generation*. For each $(q, a) \in \mathcal{S}'_{QA}$ and for each $M_{strong} \in \mathcal{M}_{\text{strong}}$, invoke our chat template C_{gen} repeatedly to produce multiple, diverse candidate reasoning traces, including normal-reasoning traces and step-reasoning traces (Section 3.3). (iii) *Step-Mask Reasoning Scoring*. Apply our proposed **Step-Mask** to each candidate reasoning trace. Concatenate each masked reasoning trace with its original question q and query M_{weak}. Binary correctness outcomes at each mask level are aggregated via a Step-Mask scoring function to measure the guidance effectiveness for each candidate (Section 3.4). (iv) *Golden Reasoning Selection*. For each (q, a), select the reasoning trace with the highest step-mask score as the Golden Reasoning r^*. For excessively long r^*, truncate it appropriately to reduce its length (Section 3.5). (v) *Fine-Tuning*. Augment $\mathcal{S}'_{QA}$ to the dataset $\mathcal{S}_{QAR} = \{(q, a, r^*)\}$ by adding $r^*$and then fine-tune M_{weak} (Section 3.6). Figure 2 also

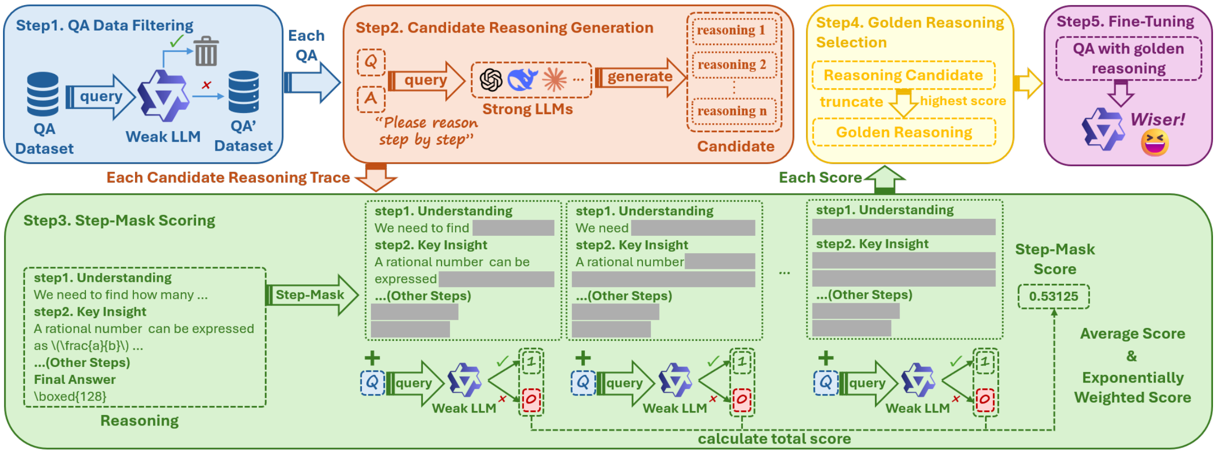

Figure 2: Overview of our proposed Weak2Wise framework. **(i)** Filter QA pairs answered incorrectly by M_{weak}. **(ii)** Use strong LLMs $\mathcal{M}_{\text{strong}}$ to generate multiple reasoning traces. **(iii)** Apply Step-Mask to score each reasoning trace. **(iv)** Select the highest-scoring reasoning trace r^* and truncate it. **(v)** Fine-tune M_{weak} on the augmented golden reasoning dataset.

illustrates our Weak2Wise framework.

3.2 Question–Answer Data Filtering

To select high-quality question-answer pairs and better facilitate subsequent steps, we first need to filter the given question-answer dataset $\mathcal{S}_{QA}$. During the filtering process, we consider questions that weak LLM M_{weak} cannot answer correctly as high-quality ones. This is because these question-answer pairs reveal that the weak LLM M_{weak} is unable to complete certain high-difficulty reasoning processes. Using these high-quality question-answer pairs to fine-tune weak LLM M_{weak} can maximize the improvement of its reasoning abilities.

Let $\mathcal{S}_{QA}$ denote the original question–answer dataset, and let M_{weak} be the base LLM with weak reasoning capability. We apply each $(q, a) \in \mathcal{S}_{QA}$ to M_{weak}. The filtered subset $\mathcal{S}'_{QA}$ retains only those pairs for which M_{weak} answers incorrectly:

$$\mathcal{S}'_{QA} = \{(q, a) \in \mathcal{S}_{QA} \mid M_{\text{weak}}(q) \neq a\} \quad (1)$$

where q is a question in $\mathcal{S}_{QA}$ and a is its ground-truth answer. The symbol "$\neq$" indicates that $M_{\text{weak}}(q)$ and a are not semantically equivalent.

To automatically determine whether $M_{\text{weak}}(q)$ and a are semantically equivalent, we designed a prompt template P_{judge} (see Figure 7 in Appendix A for details) that uses an additional LLM to automatically judge whether $M_{\text{weak}}(q)$ is correct.

3.3 Candidate Reasoning Traces Generation

Given the filtered question–answer set $\mathcal{S}'_{QA}$ (Section 3.2), our goal at this stage is to produce a diverse set of candidate reasoning traces for each question-answer pair $(q, a) \in \mathcal{S}'_{QA}$. Concretely, for each (q, a) and for each strong model $M_{\text{strong}} \in \mathcal{M}_{\text{strong}}$, we repeatedly invoke a unified chat prompt template C_{gen} (see Figure 3 for details) and collect multiple reasoning traces.

{ role: "user"; content: question}
{ role: "assistant"; content: answer}
{ role: "user"; content: "Please reason step by step. Before every step, must output a subtitle beginning with '##'. The subtitle of the last step must be '## Final Answer'."}

Figure 3: The chat template C_{gen} used for *Candidate Reasoning Traces Generation* (Section 3.3)

Chat Template C_{gen}. The chat template C_{gen} integrates the question and answer in the context of the LLM and prompts the LLM to output the reasoning trace from the question to the answer in a step-by-step format. Based on C_{gen}, $\mathcal{M}_{\text{strong}}$ will output two parts in their responses: "reasoning content" and "content". Here is a real case of "reasoning content" and "content" in Appendix D.

- **Normal-Reasoning Trace:** the "reasoning content" is the LLM's own reasoning process. It is typically characterized by multiple occurrences of "wait" to check its own reasoning process. We refer to the "reasoning content" as the Normal-Reasoning Trace, which will be used for fine-tuning.
- **Step-Reasoning Traces**: the "content" is the

step-by-step reasoning process output according to our prompt, with each step beginning with "##". We refer to the "content" as the Step-Reasoning Trace, which will be used for golden reasoning selection.

Diversity and Coverage. To ensure both breadth and depth in the candidate reasoning trace set, we apply two orthogonal diversity strategies:

- **Inter-Model Diversity**: sampling outputs from multiple $M_{\text{strong}} \in \mathcal{M}_{\text{strong}}$ captures differing reasoning styles.
- **Intra-Model Diversity**: sampling the same M_{strong} multiple times at a relatively high temperature uncovers alternative reasoning paths.

The final candidate reasoning trace set for each (q, a) is:

$$\mathcal{R}_{cand}(q, a) = \bigcup_{M \in \mathcal{M}_{\text{strong}}} \mathcal{R}_{cand}(q, a, M) \quad (2)$$

where $\mathcal{R}_{cand}(q, a, M)$ is the set of all reasoning traces sampled by M_{strong} for (q, a). Each candidate reasoning trace $r \in \mathcal{R}_{cand}(q, a)$ comprises both **Normal-Reasoning Trace** (leveraged in Section 3.5 and Section 3.6) and **Step-Reasoning Trace** (leveraged in Section 3.4).

3.4 Step-Mask Reasoning Scoring

Given the candidate reasoning traces $\mathcal{R}_{cand}(q, a)$ generated in Section 3.3, our goal in this stage is to yield a score to measure the guidance effectiveness for each reasoning trace. This score measures how effectively each reasoning trace guides the weak LLM M_{weak} toward the correct answer when partial reasoning is masked. We achieve this via our **Step-Mask** procedure, which produces a standard score $s(r)$ for each candidate reasoning trace $r \in \mathcal{R}_{cand}(q, a)$.

Step-Mask Construction. As described in Section 3.3, each reasoning trace comprises two components: the Normal-Reasoning trace and the Step-Reasoning trace. In this stage, we leverage the Step-Reasoning trace, denoted r_s. Let a Step-Reasoning trace r_s consists of K ordered steps, each encoded as a character sequence. We define a mask granularity parameter $n \in \mathbb{N}^+$. For each granularity level $i \in \{0, 1, \ldots, n-1\}$, we simultaneously mask the final $\frac{i}{n}$ fraction of characters in

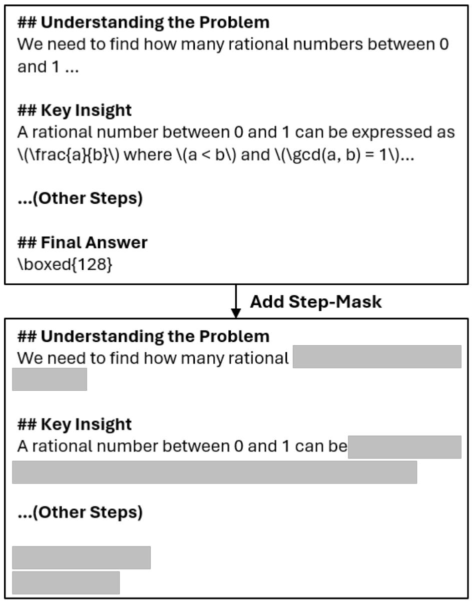

Figure 4: The process of adding a step mask to the Step-Reasoning trace (Section 3.4). The gray area represents the step-mask, which in practice is replaced with the placeholder `"(to be continued...)"`. To avoid the M_{weak} simply copying the correct answer, we fully mask the "## Final Answer" step of each trace.

> ## Problem
> {question}
>
> ## Hint
> {Step-Reasoning trace}
>
> Please reason step by step, and put your final answer within \boxed{}.

Figure 5: The prompt template P_{qr} used to concatenate Step-Reasoning trace with its original question (Section 3.4).

every step, producing n masked variants

$$r_s^{(i)} = \left(r_{s,1}^{(i)}, \ldots, r_{s,K}^{(i)}\right), \quad (3)$$

where for step k of length ℓ_k, we replace the last $\lceil \frac{i}{n} \ell_k \rceil$ characters with the placeholder `"(to be continued...)."` Figure 4 illustrates the process of adding a step mask to the Step-Reasoning trace.

Binary Correctness Evaluation. Each masked Step-Reasoning trace $r_s^{(i)}$ is concatenated with the original question q using the prompt template P_{qr} (see Figure 5 for details), and the resulting prompt is fed into M_{weak}. We then record the binary outcome:

$$s^{(i)} = \begin{cases} 1, & \text{if } M_{\text{weak}}(q \| r_s^{(i)}) = a, \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

where the symbol "=" indicates that $M_{\text{weak}}(q \| r_s^{(i)})$ and a are semantically equivalent. We continue to use the prompt template P_{judge} (see Figure 7 in Appendix A for details) to automate this evaluation. As a result, each Step-Reasoning trace produces a set of n binary scores: $\{s^{(0)}, s^{(1)}, \ldots, s^{(n-1)}\}$.

Step-Mask Scoring Function. From these binary outcomes, we compute two complementary metrics:

- **Average Step-Mask Score**

$$s_{\text{avg}}(r) = \frac{1}{n} \sum_{i=0}^{n-1} s^{(i)}, \tag{5}$$

which captures the overall guidance effectiveness of r under varying mask strengths.

- **Exponentially Weighted Step-Mask Score**

$$s_{\text{ew}}(r) = \frac{1}{n-1} \sum_{i=1}^{n-1} s^{(i)} \cdot 2^{-(n-i)} + s^{(0)} \cdot 2^{-(n-1)}, \tag{6}$$

which assigns greater weight to success under heavier masking, emphasizing the reasoning structure's robustness and guidance effectiveness.

Finally, we combine these into a single step-mask score to measure guidance effectiveness:

$$s(r) = \beta \cdot s_{\text{avg}}(r) + (1-\beta) \cdot s_{\text{ew}}(r), \tag{7}$$

where $\beta \in [0, 1]$ is a tunable hyperparameter balancing overall quality and structural quality of the reasoning trace r. The step-mask scores $s(r)$ are then used in Section 3.5 to select high-quality reasoning traces.

In the ablation studies (Section 4.3.1), we will demonstrate the efficacy of this step-mask scoring function as a metric for evaluating the quality of reasoning traces.

3.5 Golden Reasoning Selection

Having computed a standard step-mask score $s(r)$ for each candidate reasoning trace $r \in \mathcal{R}_{cand}(q, a)$ (Section 3.4), our goal is this stage is to select and, if necessary, truncate the optimal reasoning trace to serve as the Golden Reasoning r^*. This stage comprises two steps: *selection* over all candidates, and *length-aware truncation* to enforce practical constraints on trace size.

Selection. For each $(q, a) \in \mathcal{S}'_{QA}$, we select the candidate reasoning trace whose step-mask score is maximal:

$$r^* = \arg\max_{r \in \mathcal{R}_{cand}(q,a)} s(r). \tag{8}$$

In the event that multiple reasoning traces achieve the maximal step-mask score, we select the reasoning trace with the fewest tokens, since such concise reasoning traces maintain equal guidance effectiveness while being easier for the weak LLM M_{weak} to comprehend.

Length-Aware Truncation. During fine-tuning, we utilize the Normal-Reasoning trace portion of r^* rather than the Step-Reasoning trace, since the latter is overly abstract for M_{weak} and thus difficult to internalize. However, Normal-Reasoning traces often include repeated “wait” backtracks after the correct answer has already been found. To eliminate this redundancy, we introduce a prompt template P_{trunc} (see Figure 8 for details) that automatically truncates excessively long Normal-Reasoning trace at the first occurrence of the correct answer. The resulting truncated trace $\widetilde{r}^*$ is significantly shorter and thus better suited for downstream fine-tuning of M_{weak}. Here is a real case of a Normal-Reasoning Trace before and after truncation in Appendix D.

Resulting Dataset. After selection and truncation, we obtain the final Golden Reasoning r^* (or $\widetilde{r}^*$ if truncated) for each (q, a). We then form the enriched dataset

$$\mathcal{S}_{QAR} = \{ (q, a, r^*) \mid (q, a) \in \mathcal{S}'_{QA} \}, \tag{9}$$

which serves as the basis for fine-tuning the weak model M_{weak} in Section 3.6.

In the ablation studies (Section 4.3.2), we will demonstrate the efficacy of truncating reasoning trace for downstream fine-tuning.

3.6 Fine-Tuning

In the final stage, we perform supervised fine-tuning of the weak model M_{weak} on the enriched dataset $\mathcal{S}_{QAR}$. Each training example is formatted as Figure 6, where $(q, a, r^*) \in \mathcal{S}_{QAR}$.

During optimization, we encourage the model to generate both the reasoning and the answer in sequence, and compute the loss on reasoning and answer tokens. Let $\mathbf{y} = (y_1, \ldots, y_T)$ denote the full sequence of response tokens and i the index of

prompt: $\{q\}$
response: <think> $\{r^*\}$ </think> $\{a\}$

Figure 6: The concatenation format of fine-tuning training data (Section 3.6).

the "<think>" marker. The supervised fine-tuning loss function is:

$$\mathcal{L}(\theta) = -\frac{1}{|\mathcal{S}_{QAR}|} \sum_{(q,a,r^*)\in\mathcal{S}_{QAR}} \sum_{t=i}^{T} \log p_\theta(y_t \mid q,\, y_{<t}). \tag{10}$$

By masking out the question part during gradient computation, M_{weak} is guided to assimilate $\mathcal{M}_{\text{strong}}$'s reasoning capabilities while preserving focus on reasoning and answer.

4 Experiments

4.1 Experiment Setup

Dataset Synthesis. We apply our Weak2Wise framework for synthesizing high-quality reasoning traces from existing QA pairs in S1K dataset (Muennighoff et al., 2025). It is worth noting that S1K is a high-quality math and science QA dataset with 1,000 samples, which already excludes questions answerable by Qwen2.5-7B. Thus, the *Data Filtering* step in Weak2Wise can be skipped in our experiments. We adopt DeepSeek r1 (Guo et al., 2025) and QwQ-Plus (Yang et al., 2024) as $\mathcal{M}_{\text{strong}}$ and Qwen2.5-7B as M_{weak}. Each model in $\mathcal{M}_{\text{strong}}$ samples 3 distinct reasoning traces per question with $temperature = 0.3$, producing a diverse candidate pool. We then set the mask granularity parameter $n = 6$ and the score weight parameter $\beta = 0.5$ in Step-Mask Scoring stage. DeepSeek-V3 (Liu et al., 2024) is leveraged for correctness judgment and reasoning trace truncation with $temperature = 0.1$. We use the "majority vote" principle assessing each answer three times to ensure reliability. After incorporating golden reasoning into the S1K dataset via Weak2Wise, we obtained the **GR-1K** dataset.

Training. We perform supervised finetuning on Qwen2.5-7B with our GR-1K dataset to obtain our model **GR-7B**. Detailed training hyper-parameters can be found in Appendix B

Evaluation. Following Muennighoff et al., 2025, we evaluate GR-7B and other models on 3 challenging reasoning benchmarks: AIME2024 (Maxwell-Jia, 2024), MATH-500 (Lightman et al., 2023) and GPQA Diamond (Rein et al., 2023). The questions in all benchmarks are not present in GR-1K. The metrics are scores of these benchmarks.

- **AIME2024:** 30 three-digit answer math problems from the 2024 American Invitational Mathematics Examination.
- **MATH-500:** A set of 500 college-level competition questions covering algebra, geometry, number theory, and probability.
- **GPQA Diamond:** 198 graduate-level science questions spanning biology, chemistry, and physics qualifiers.

Other Models. Following Cai et al., 2025, we compare **GR-7B** against five competitive 7B-parameter models: LLaMA-o1 trained on 332K reasoning examples (SimpleBerry, 2025), Macro-o1 trained on 60K reasoning examples (Zhao et al., 2024), Bespoke-Stratos-7B trained on 17K distilled chains (Bespoke Labs, 2025), CRV-SFT-7B trained on 17K distilled chains (Cai et al., 2025) and S1.1-7B trained on 1K DeepSeek-R1 reasoning traces (Muennighoff et al., 2025). We evaluate all models under “lm-evaluation-harness” framework (Biderman et al., 2024) to ensure a fair comparison.

4.2 Main Results

To contextualize GR-7B's performance among other competitive 7B-parameter models, Table 2 compares accuracy and training set sizes. Despite being trained on only 1K reasoning traces, GR-7B outperforms larger-data baselines such as LLaMA-o1 (332K examples) and Macro-o1 (60K examples), as well as distilled-chain models Bespoke-Stratos-7B and CRV-SFT-7B (17K examples each). Notably, GR-7B achieves the highest accuracies on all benchmarks—26.7% on AIME2024, 84.2% on MATH-500, and 42.4% on GPQA Diamond—while matching S1.1-7B's minimal training set size. This highlights the efficiency and effectiveness of our Weak2Wise approach in reasoning synthesis.

Table 2 also reports the performance of our fine-tuned model GR-7B against its backbone Qwen2.5-7B on three standard reasoning benchmarks. Compared to Qwen2.5-7B, GR-7B achieves substantial gains in accuracy across all datasets: an absolute increase of 16.7% accuracy on AIME2024

Model	Training Set Size	AIME2024	MATH-500	GPQA Diamond
LLaMA-o1	332K	3.3	28.6	26.3
Macro-o1	60K	6.7	38.4	31.8
Bespoke-Stratos-7B	17K	20.0	82.0	37.8
CRV-SFT-7B	17K	20.0	80.0	37.4
S1.1-7B	1K	20.0	83.0	40.9
Qwen2.5-7B(backbone)	0	10.0	73.6	33.3
GR-7B	**1K**	**26.7**	**84.2**	**42.4**

Table 2: Performance comparison of various 7B-parameter models on AIME2024, MATH-500, and GPQA Diamond benchmarks

(from 10.0% to 26.7%), 10.6% on MATH-500 (from 73.6% to 84.2%), and 9.1% on GPQA Diamond (from 33.3% to 42.4%). These results demonstrate that incorporating high-quality, weak-LLM-Friendly reasoning traces via Weak2Wise substantially enhances the reasoning capabilities of Qwen2.5-7B even with only 1K fine-tuning examples.

4.3 Ablation Studies

To further validate the contributions of our Step-Mask Scoring and reasoning trace truncation, we conduct two complementary ablation experiments. Table 3 compares the effect of different candidate selection strategies, and Table 4 examines the impact of disabling our truncation step.

4.3.1 Effectiveness of Step-Mask Scoring

In the first ablation (Table 3), we replace our highest-scoring trace selection with (i) random sampling, (ii) lowest Step-Mask Score, and (iii) highest Step-Mask Score (our full method). We then fine-tune Qwen2.5-7B on each resulting dataset. Selecting reasoning traces at random yields a substantial drop in accuracy, and using the lowest-scoring traces performs even worse than the backbone model on MATH-500 and GPQA Diamond. In contrast, choosing the top-ranked reasoning traces via our Step-Mask Scoring consistently delivers the best results across all benchmarks. This demonstrates that our scoring metric effectively quantifies reasoning trace quality and that prioritizing high-scoring candidates is crucial for reliable supervision. We additionally conduct a statistical analysis of the scores generated by Step-Mask Scoring in Appendix C.

4.3.2 Impact of Reasoning Trace Truncation

The second ablation (Table 4) evaluates the effect of disabling our truncation step. Without truncation, accuracy decreases markedly. We attribute this degradation to QwQ-Plus occasionally generating reasoning traces longer than 32,768 tokens, which exceed Qwen2.5-7B's context window. Incorporating these extremely-long reasoning traces during fine-tuning leads to incomplete reasoning and flawed inference. By truncating traces properly to fit within the model's context capacity, we preserve the integrity of the learned reasoning patterns and achieve substantial performance gains.

Dataset	random	lowest	GR-7B
AIME2024	16.7	10.0	**26.7**
MATH-500	78.2	70.6	**84.2**
GPQA Diamond	39.4	30.8	**42.4**

Table 3: Ablation on candidate selection: random vs. lowest-scoring vs. GR-7B (highest-scoring traces).

Dataset	w/o Truncation	GR-7B
AIME2024	16.7	**26.7**
MATH-500	72.2	**84.2**
GPQA Diamond	35.4	**42.4**

Table 4: Ablation on truncation: without truncation vs. GR-7B (with truncation).

5 Conclusion

In this paper, we introduce Weak2Wise, a fully automated, lightweight framework for synthesizing high-quality, weak-LLM-friendly reasoning traces. Our framework innovates by leveraging the weak LLM's own performance to evaluate reasoning traces with a novel step-mask scoring mechanism. Through experiments on challenging benchmarks, we demonstrated that Weak2Wise effectively synthesize reasoning traces for fine-tuning weak LLMs. Weak2Wise offers an efficient solution for improving reasoning in LLMs, making high-quality reasoning more accessible for practical applications.

Limitations

While Weak2Wise has demonstrated promising results, it is important to acknowledge some limitations inherent in its current design. One notable aspect is its reliance on data distillation from strong LLMs, which is a common limitation of data distillation approach. Although this approach has been carefully implemented with multiple sampling and selection processes to ensure robustness, the performance of Weak2Wise remains closely tied to the quality of the strong LLMs used. There may be instances where the reasoning traces, despite our efforts to optimize their selection, could still contain subtle deficiencies or biases resulting from the limitations in strong LLMs. These factors might influence the learning effectiveness of weak LLMs. Future work will focus on exploring additional strategies to further mitigate such potential limitations and enhance the overall robustness and independence of the Weak2Wise framework.

Ethical Considerations

In conducting our research, we have thoroughly reviewed and ensured compliance with ethical standards. Our study utilizes existing datasets, which have been publicly available and previously vetted for ethical use. These datasets have been carefully selected to avoid any form of offensive or biased content. Therefore, we consider that our research does not present any ethical issues. The data used is ethically sourced, the analysis is unbiased, and all procedures align with established ethical guidelines.

References

Bespoke Labs. 2025. Bespoke-Stratos-7B: A Fine-tuned Model for Math Reasoning. https://huggingface.co/bespokelabs/Bespoke-Stratos-7B. Accessed: 2025-05-17.

Milan Bhan, Jean-Noël Vittaut, Nicolas Chesneau, and Marie-Jeanne Lesot. 2024. Self-amplify: Improving small language models with self post hoc explanations. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10974–10991.

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, and 1 others. 2024. Lessons from the trenches on reproducible evaluation of language models. *arXiv preprint arXiv:2405.14782*.

Wenrui Cai, Chengyu Wang, Junbing Yan, Jun Huang, and Xiangzhong Fang. 2025. Training small reasoning llms with cognitive preference alignment. *arXiv preprint arXiv:2504.09802*.

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*.

Fatemeh Haji, Mazal Bethany, Maryam Tabar, Jason Chiang, Anthony Rios, and Peyman Najafirad. 2024. Improving llm reasoning with multi-agent tree-of-thought validator agent. *arXiv preprint arXiv:2409.11527*.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2023. Large language models are reasoning teachers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14852–14882.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8003–8017, Toronto, Canada. Association for Computational Linguistics.

Linmei Hu, Hongyu He, Duokang Wang, Ziwang Zhao, Yingxia Shao, and Liqiang Nie. 2024. Llm vs small model? large language model based text augmentation enhanced personality detection model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18234–18242.

Minki Kang, Seanie Lee, Jinheon Baek, Kenji Kawaguchi, and Sung Ju Hwang. 2023. Knowledge-augmented reasoning distillation for small language models in knowledge-intensive tasks. *Advances in Neural Information Processing Systems*, 36:48573–48602.

Yujin Kim, Euiin Yi, Minu Kim, Se-Young Yun, and Taehyeon Kim. 2025. Guiding reasoning in small language models with llm assistance. *arXiv preprint arXiv:2504.09923*.

Chenglin Li, Qianglong Chen, Liangyue Li, Caiyu Wang, Yicheng Li, Zulong Chen, and Yin Zhang. 2023a. Mixed distillation helps smaller language model better reasoning. *arXiv preprint arXiv:2312.10730*.

Liunian Harold Li, Jack Hessel, Youngjae Yu, Xiang Ren, Kai-Wei Chang, and Yejin Choi. 2023b. Symbolic chain-of-thought distillation: Small models can also “think” step-by-step. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2665–2679.

Zhiming Li, Yushi Cao, Xiufeng Xu, Junzhe Jiang, Xu Liu, Yon Shin Teo, Shang-Wei Lin, and Yang Liu. 2024. Llms for relational reasoning: How far are we? In *Proceedings of the 1st International Workshop on Large Language Models for Code*, pages 119–126.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

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*.

Hanmeng Liu, Zhiyang Teng, Leyang Cui, Chaoli Zhang, Qiji Zhou, and Yue Zhang. 2023. Logicot: Logical chain-of-thought instruction tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2908–2921.

Alisia Lupidi, Carlos Gemmell, Nicola Cancedda, Jane Dwivedi-Yu, Jason Weston, Jakob Foerster, Roberta Raileanu, and Maria Lomeli. 2024. Source2synth: Synthetic data generation and curation grounded in real data sources. *arXiv preprint arXiv:2409.08239*.

Maxwell-Jia. 2024. Aime 2024 dataset.

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*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Synthetic prompting: Generating chain-of-thought demonstrations for large language models. In *International Conference on Machine Learning*, pages 30706–30775. PMLR.

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.

SimpleBerry. 2025. LLaMA-O1: Open Large Reasoning Model Frameworks For Training, Inference and Evaluation With PyTorch and HuggingFace. `https://github.com/SimpleBerry/LLaMA-O1`. Accessed: 2025-05-17.

Peifeng Wang, Zhengyang Wang, Zheng Li, Yifan Gao, Bing Yin, and Xiang Ren. 2023. Scott: Self-consistent chain-of-thought distillation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5546–5558.

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, 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.

Yuanhao Yue, Chengyu Wang, Jun Huang, and Peng Wang. 2024. Distilling instruction-following abilities of large language models with task-aware curriculum planning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6030–6054, Miami, Florida, USA. Association for Computational Linguistics.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D Goodman. 2024. Star: Self-taught reasoner bootstrapping reasoning with reasoning. In *Proc. the 36th International Conference on Neural Information Processing Systems*, volume 1126.

Biao Zhang, Zhongtao Liu, Colin Cherry, and Orhan Firat. 2024. When scaling meets llm finetuning: The effect of data, model and finetuning method. In *The Twelfth International Conference on Learning Representations*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*.

Yu Zhao, Huifeng Yin, Bo Zeng, Hao Wang, Tianqi Shi, Chenyang Lyu, Longyue Wang, Weihua Luo, and Kaifu Zhang. 2024. Marco-o1: Towards open reasoning models for open-ended solutions. *arXiv preprint arXiv:2411.14405*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, and 1 others. 2023. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36:55006–55021.

Xuekai Zhu, Biqing Qi, Kaiyan Zhang, Xinwei Long, Zhouhan Lin, and Bowen Zhou. 2024. Pad: Program-aided distillation can teach small models reasoning better than chain-of-thought fine-tuning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2571–2597.

A Prompt Template

Here are all the prompt templates used in Section 3. Each prompt template has been custom-designed to automate the entire workflow without any manual intervention.

> You're an AI evaluator for science questions. The user will give you a question, an attempt and the correct answer.
>
> Your task is to compare the attempt with the provided correct answer and determine whether it is correct. If the correct answer is a clear numerical value or a multiple-choice option, there must be no ambiguity. If the correct answer requires a full reasoning process, assess whether the attempt is valid, using the correct answer as a reference if necessary.
>
> The user will supply the input in the following format:
> ## Question
> {question}
> ## Attempt
> {attempt to be evaluated}
> ## Correct Answer
> {correct answer}
>
> Explain your evaluation step by step, and finish your response on a new line with only "Yes" or "No".

Figure 7: The prompt template P_{judge} used to determine the correctness of LLM's response to certain question. (Section 3.2 and Section 3.4)

> You are a helpful assistant who is highly skilled at simplifying reasoning processes. User will provide you with the reasoning process and correct answer for a certain question. There are many backtracking in reasoning, which always start with the word 'wait'.
> You need to simplify the reasoning process in the following way: Extract from the beginning of the reasoning until the correct answer is FIRST deduced. Note that all you need to do is find the appropriate endpoint and output the reasoning process from the beginning to the endpoint. No modification of any reasoning content is allowed.
>
> Just output the simplified reasoning process without any additional content.
>
> Here is the reasoning process and correct answer for a certain question from the user:
>
> ## Reasoning Process
> {Normal-Reasoning trace}
> ## Correct Answer
> {answer}

Figure 8: The prompt template P_{trunc} used to truncate excessively long Normal-Reasoning traces (Section 3.5).

B Training Details

We fine-tune the Qwen2.5-7B model on our GR-1K dataset. We train for 3 epochs with a learning rate of 1e-5, using a global batch size of 16. Packing is enabled to optimize GPU memory usage by combining multiple shorter sequences. A cosine learning rate scheduler is applied with a warm-up ratio of 0.03 and weight decay of 0.01 to balance training stability and convergence. The training takes approximately 2 hours on an 8 NVIDIA A100 GPU server.

C Further Experiments

Here, we additionally conduct a statistical analysis of the scores generated by Step-Mask Scoring (Section 3.4) for each reasoning. From the **Step-Mask Scoring Function** in Section 3.4, it can be easily proven that the value of the step-mask score ranges between 0 and 1. The step-mask score is 0 when all $s^{(i)}$ are 0, and 1 when all $s^{(i)}$ are 1.

Specifically, we analyze the distribution of the ranges of $s^{(i)}$ generated by each reasoning, which aims to demonstrate that our step-mask scoring design has strong separability for different reasoning traces. From Figure 9 (left), We observe that in only a small proportion of reasoning traces, all $s^{(i)}$ exhibit a range of less than 0.1 (10% of the total score range of 0–1). This indicates that our step-mask scoring reflects differences across different reasoning traces, demonstrating strong separability.

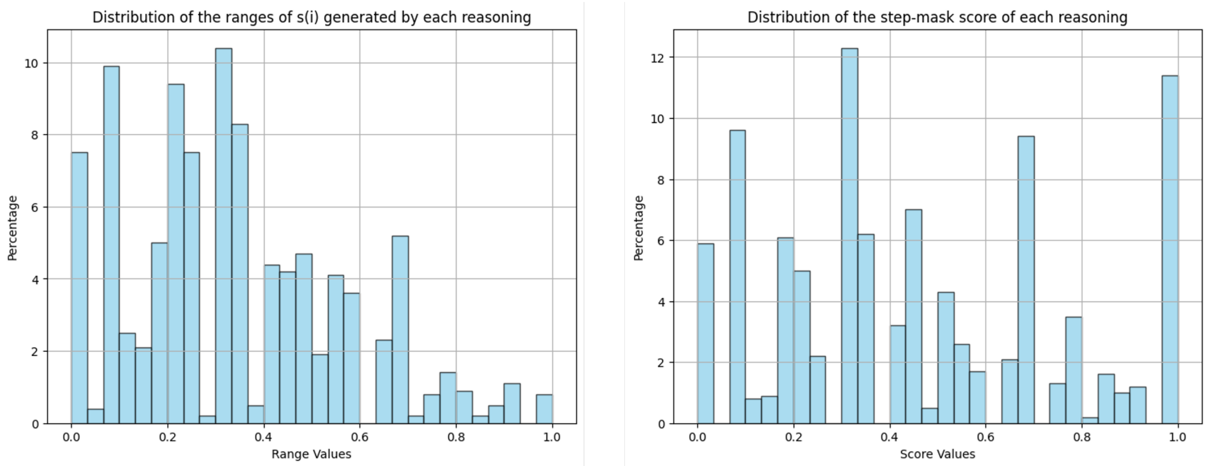

Figure 9: Distribution of the ranges of $s^{(i)}$ generated by each reasoning (left) And Distribution of the step-mask scores of each reasoning (right).

We additionally analyze the scores distribution of all golden reasoning, which correspond to the highest scores among all reasoning traces for each QA pair. We find that these scores are distributed across the entire 0–1 range. We believe this reflects a uniform distribution of problem difficulties in our selected QA set. For simple questions, strong LLMs can generate high-quality reasoning traces (with scores closer to 1), whereas for difficult questions, even strong LLMs may only produce lower-quality ones (with scores closer to 0), despite our selection process.

It is particularly notable that 6% of the golden reasoning traces scored 0, which may be ineffective for fine-tuning weak LLMs and enhancing their reasoning abilities. For these golden reasoning traces and their corresponding QA pairs, we propose the following remedies: (i) discard them to prevent weak LLMs from learning low-quality or incorrect reasoning processes; (ii) resample by having strong LLMs generate more reasoning traces for the corresponding QA pairs until a reasoning trace with a score higher than 0 is obtained; (iii) manually annotate them. Given the dataset's overall size of approximately 1K, 6% translates to only around 60 data points, which is a manageable task. Moreover, these QA pairs may also be too difficult for strong LLMs to answer properly, so manual annotation would better improve data quality.

Finally, we analyze which strong LLM our selected golden reasoning traces originated from. In our experiments, 56.70% of the golden reasoning traces come from DeepSeek r1 and 43.3% from QwQ-Plus. We believe these proportions reflect the weak LLM's preference for different strong LLMs. To put it more vividly, they show which strong LLM the weak LLM tends to select as its "teacher".

D Case Studies

Here is a real case of golden reasoning synthesis from the original QA. The data in the Question and Answer columns were extracted from the original QA dataset S1K (Muennighoff et al., 2025). After applying the chat template C_{gen} in Figure 3, the strong LLM generated the corresponding Normal-Reasoning Trace ("reasoning content") and Step-Reasoning Trace ("content"). We performed Step-Mask Scoring on the Step-Reasoning Trace and found that it had the highest score among all generated reasoning traces. Subsequently, due to the excessive length of its Normal-Reasoning Trace, we used P_{trunc} in Figure

8 for truncation, and ultimately obtained the golden reasoning for fine-tuning. We mark the complete Normal-Reasoning Trace with light blue and the truncated Normal-Reasoning Trace with dark blue.

Question

Given a rational number, write it as a fraction in lowest terms and calculate the product of the resulting numerator and denominator. For how many rational numbers between 0 and 1 will $20!$ *be the resulting product?*

Correct Answer

128

Normal-Reasoning Trace ("reasoning content")

Okay, so the problem is asking: For how many rational numbers between 0 and 1 will 20! be the product of the numerator and denominator when the rational number is written in lowest terms. Hmm, let me try to unpack this step by step.

First, let me make sure I understand the question correctly. We need to find all fractions between 0 and 1 that, when reduced to their simplest form (i.e., numerator and denominator are coprime), the product of the numerator and denominator equals 20 factorial (20!). Then, count how many such fractions exist.

So, to rephrase, we are looking for pairs of integers (a, b) such that:

1. $0 < a < b$ *(since the fraction is between 0 and 1)*

2. $\gcd(a, b) = 1$ *(the fraction is in lowest terms)*

3. $a \cdot b = 20!$ *(the product of numerator and denominator is 20!)*

Therefore, the task reduces to finding the number of coprime pairs (a, b) with $0 < a < b$ *and* $a \cdot b = 20!$*.*

Hmm, how do I approach this? Let me think. Since a and b are coprime and their product is 20!, they must form a coprime pair of divisors of 20!. So, essentially, we need to count the number of ways to split the prime factors of 20! into two coprime parts a and b, such that $a < b$*.*

But since a and b are coprime, they must share no common prime factors. That means all the prime factors of 20! have to be partitioned between a and b, with no overlap. So, each prime factor in 20! must go entirely to a or entirely to b.

Wait, that makes sense. Because if a prime p *divides both a and b, then* $\gcd(a, b)$ *would be at least* p*, contradicting the coprimality. Therefore, each prime in the prime factorization of 20! must be assigned exclusively to either a or b.*

Therefore, the problem becomes equivalent to finding the number of ways to partition the set of prime factors of 20! into two non-empty subsets (since a and b must both be at least 1) such that the product of the primes in each subset (with their respective exponents) gives a and b, and then ensuring that $a < b$*.*

But wait, actually, it's not just the primes, but all the prime powers in the factorization of 20!. Because 20! has exponents for each prime. For example, $20! = 2^{18} \cdot 3^8 \cdot 5^4 \cdot 7^2 \cdot 11^1 \cdot 13^1 \cdot 17^1 \cdot 19^1$ *(let me check that later). So, each prime power must be entirely assigned to either a or b.*

But since a and b are coprime, as long as all occurrences of a prime are assigned to one side, they can be. So, the process is similar to assigning each prime in the prime factorization to either a or b. For each distinct prime, we have two choices: assign all its exponents to a or to b. However, since the order matters $(a < b)$*, but we have to account for that.*

But wait, the number of coprime pairs (a, b) with $a \cdot b = N$ *is* 2^{k-1}*, where* k *is the number of distinct prime factors of* N*. Because for each prime, you assign it to a or b, giving* 2^k *possibilities. However, since a and b are positive integers and* $a \leq b$*, we divide by 2, but since* N *is a square only if there's a square factor, but in our case, since a and b are coprime,* N *cannot be a square unless all exponents are even, but 20! has exponents which are not all even. Wait, but 20! is not a perfect square. Let me check.*

Wait, 20! includes primes like 2, which has exponent 18 (even), 3 has exponent 8 (even), 5^4 *(even),* 7^2 *(even), but 11, 13, 17, 19 each have exponent 1 (odd). So, 20! is not a perfect square. Therefore, there are no solutions where* $a = b$*. Therefore, the total number of unordered pairs* $\{a, b\}$ *with* $a \cdot b = N$ *and* $\gcd(a, b) = 1$ *is* 2^k*, where* k *is the number of distinct prime factors, divided by 2 (since unordered), which would be* 2^{k-1}*. But since we need ordered pairs (a, b) with* $a < b$*, the number would be* 2^{k-1}*.*

But hold on, in our problem, N is 20!, so k is the number of distinct prime factors of 20!.

Let me first compute the prime factors of 20!.

Primes less than or equal to 20 are: 2, 3, 5, 7, 11, 13, 17, 19. That's 8 primes. Therefore, $k = 8$. Therefore, the number of coprime pairs (a, b) with $a \cdot b = 20!$ would be $2^{8-1} = 128$. But wait, is that correct?

(The first occurrence of the correct answer)

Wait, the standard formula for the number of coprime pairs (a, b) with $a \cdot b = N$ is 2^k, where k is the number of distinct prime factors of N. Because each prime can go to a or b. Since a and b must be positive integers, and each coprime decomposition corresponds to a subset of primes assigned to a, and the rest to b. Therefore, the number of such pairs is 2^k. However, since the problem requires $a < b$, then we need to divide by 2, but only if $a \neq b$. Since N is not a perfect square (as established earlier), all pairs have $a \neq b$. Therefore, the number of ordered pairs (a, b) with $a < b$ and $a \cdot b = N$ and $\gcd(a, b) = 1$ is 2^{k-1}.

Therefore, since $k = 8$, the number would be $2^7 = 128$. Therefore, the answer would be 128.

But wait, let me verify this carefully. Let's take a smaller example. Suppose N is 6, which is $2 \cdot 3$. Then, the coprime pairs (a, b) with $a \cdot b = 6$ are (1,6) and (2,3). However, (1,6) and (6,1), (2,3) and (3,2). But since $a < b$, we have two pairs: (1,6) and (2,3). The number of distinct prime factors of 6 is 2 (primes 2 and 3). So, $2^{2-1} = 2$, which matches. Similarly, for $N = 30$ (primes 2, 3, 5), $k = 3$, so $2^2 = 4$. Let's see: the coprime pairs with $a \cdot b = 30$ and $a < b$ are (1,30), (2,15), (3,10), (5,6). That's 4 pairs. Yep, that works. So the formula holds.

Therefore, applying this to 20!, which has 8 distinct prime factors, the number should be $2^7 = 128$. Hence, the answer is 128.

But hold on a second. Let me check the prime factors of 20! again to confirm the count of distinct primes. The primes less than or equal to 20 are 2, 3, 5, 7, 11, 13, 17, 19. That's 8 primes. So $k = 8$. Thus, $2^{8-1} = 128$. So that should be the answer.

But let me think again. Is there any possibility that in the case of 20! being a factorial, its prime exponents might affect this count? For example, does the multiplicity of primes (like 2^{18}, 3^8, etc.) play any role here? Wait, in the formula, we only consider the number of distinct primes. Because even if a prime has a higher exponent, when we split the prime factors between a and b, all the exponents of that prime must go to one side or the other. Therefore, each prime is treated as a single entity when partitioning. So, whether a prime is raised to the 1st power or the 18th power, it's still just one prime in terms of assignment. Hence, the number of coprime pairs is determined solely by the number of distinct primes, not their exponents. Therefore, even with multiple exponents, the number of coprime pairs remains 2^k, with k being the number of distinct primes.

Therefore, in our case, since there are 8 distinct primes in 20!, the number of coprime pairs (a, b) with $a \cdot b = 20!$ is $2^8 = 256$. However, since we require $a < b$, we divide by 2, giving 128. That seems correct. Therefore, the answer is 128. So I should check the problem statement once again to make sure I didn't miss any details.

The problem says: "how many rational numbers between 0 and 1 will 20! be the resulting product". So, each such fraction $\frac{a}{b}$, reduced to lowest terms, with $a \cdot b = 20!$. Since a and b are coprime, their product is 20!, and each such pair corresponds to a unique fraction. The count is the number of such fractions, which is the same as the number of coprime pairs (a, b) with $a < b$ and $a \cdot b = 20!$.

Therefore, yes, the answer is 128.

But let me make sure with another example. Take $N = 12$, which is $2^2 \cdot 3$. The distinct primes are 2 and 3, so $k = 2$. Then, the number of coprime pairs (a, b) with $a < b$ and $a \cdot b = 12$ is $2^{2-1} = 2$. Let's check. The possible pairs:

Possible coprime pairs (a, b) with $a \cdot b = 12$:

$1 \cdot 12$*:* $\gcd(1, 12) = 1$

$3 \cdot 4$*:* $\gcd(3, 4) = 1$

But $2 \cdot 6$*:* $\gcd(2, 6) = 2 \neq 1$

So, only two pairs: (1,12) and (3,4). So, 2 pairs, which is $2^{2-1} = 2$. Correct.

Another example: $N = 30$, as before, primes 2,3,5: 3 primes. So $2^{3-1} = 4$, which matches the four pairs. So, the formula holds. Therefore, for $N = 20!$ with 8 primes, the answer is $2^{8-1} = 128$.

Therefore, I think that's the correct answer. I don't see any mistakes in the reasoning. The key insight is that each prime must be entirely assigned to a or b, leading to 2^8 possibilities, and since we want $a < b$, we divide by 2, getting $2^7 = 128$.

Step-Reasoning Trace ("content")

Understanding the Problem

We need to find how many rational numbers between 0 and 1, when written in lowest terms, have a product of numerator and denominator equal to $20!$.

Key Observations

1. The rational number $\frac{a}{b}$ *must satisfy* $0 < a < b$ *and* $\gcd(a, b) = 1$.

2. The product $a \cdot b = 20!$.

3. a *and* b *must be coprime divisors of* $20!$.

Prime Factorization of $20!$

$20!$ *has prime factors: 2, 3, 5, 7, 11, 13, 17, 19. These are 8 distinct primes.*

Coprime Divisor Pairs

Each prime factor must be assigned entirely to either a *or* b*. For each prime, there are 2 choices (assign to* a *or* b*), leading to* $2^8 = 256$ *total pairs. Since* $a < b$*, we divide by 2 to avoid double-counting.*

Final Calculation

The number of valid pairs is $2^{8-1} = 2^7 = 128$.

Final Answer

$\boxed{128}$

Truncated Reasoning Trace for Fine-Tuning (Golden Reasoning)

Okay, so the problem is asking: For how many rational numbers between 0 and 1 will 20! be the product of the numerator and denominator when the rational number is written in lowest terms. Hmm, let me try to unpack this step by step.

First, let me make sure I understand the question correctly. We need to find all fractions between 0 and 1 that, when reduced to their simplest form (i.e., numerator and denominator are coprime), the product of the numerator and denominator equals 20 factorial (20!). Then, count how many such fractions exist.

So, to rephrase, we are looking for pairs of integers (a, b) such that:

1. $0 < a < b$ *(since the fraction is between 0 and 1)*

2. $\gcd(a, b) = 1$ *(the fraction is in lowest terms)*

3. $a \cdot b = 20!$ *(the product of numerator and denominator is 20!)*

Therefore, the task reduces to finding the number of coprime pairs (a, b) with $0 < a < b$ *and* $a \cdot b = 20!$.

Hmm, how do I approach this? Let me think. Since a and b are coprime and their product is 20!, they must form a coprime pair of divisors of 20!. So, essentially, we need to count the number of ways to split the prime factors of 20! into two coprime parts a and b, such that $a < b$.

But since a and b are coprime, they must share no common prime factors. That means all the prime factors of 20! have to be partitioned between a and b, with no overlap. So, each prime factor in 20! must go entirely to a or entirely to b.

Wait, that makes sense. Because if a prime p *divides both a and b, then* $\gcd(a, b)$ *would be at least* p*, contradicting the coprimality. Therefore, each prime in the prime factorization of 20! must be assigned exclusively to either a or b.*

Therefore, the problem becomes equivalent to finding the number of ways to partition the set of prime factors of 20! into two non-empty subsets (since a and b must both be at least 1) such that the product of the primes in each subset (with their respective exponents) gives a and b, and then ensuring that $a < b$.

But wait, actually, it's not just the primes, but all the prime powers in the factorization of 20!. Because 20! has exponents for each prime. For example, $20! = 2^{18} \cdot 3^8 \cdot 5^4 \cdot 7^2 \cdot 11^1 \cdot 13^1 \cdot 17^1 \cdot 19^1$ *(let me check that later). So, each prime power must be entirely assigned to either a or b.*

But since a and b are coprime, as long as all occurrences of a prime are assigned to one side, they can be. So, the process is similar to assigning each prime in the prime factorization to either a or b. For each

distinct prime, we have two choices: assign all its exponents to a or to b. However, since the order matters ($a < b$), but we have to account for that.
But wait, the number of coprime pairs (a, b) with $a \cdot b = N$ is 2^{k-1}, where k is the number of distinct prime factors of N. Because for each prime, you assign it to a or b, giving 2^k possibilities. However, since a and b are positive integers and $a \leq b$, we divide by 2, but since N is a square only if there's a square factor, but in our case, since a and b are coprime, N cannot be a square unless all exponents are even, but 20! has exponents which are not all even. Wait, but 20! is not a perfect square. Let me check. Wait, 20! includes primes like 2, which has exponent 18 (even), 3 has exponent 8 (even), 5^4 (even), 7^2 (even), but 11, 13, 17, 19 each have exponent 1 (odd). So, 20! is not a perfect square. Therefore, there are no solutions where $a = b$. Therefore, the total number of unordered pairs $\{a, b\}$ with $a \cdot b = N$ and $\gcd(a, b) = 1$ is 2^k, where k is the number of distinct prime factors, divided by 2 (since unordered), which would be 2^{k-1}. But since we need ordered pairs (a, b) with $a < b$, the number would be 2^{k-1}.
But hold on, in our problem, N is 20!, so k is the number of distinct prime factors of 20!.
Let me first compute the prime factors of 20!.
Primes less than or equal to 20 are: 2, 3, 5, 7, 11, 13, 17, 19. That's 8 primes. Therefore, $k = 8$. Therefore, the number of coprime pairs (a, b) with $a \cdot b = 20!$ would be $2^{8-1} = 128$.

E License For Artifacts

In our experiments, we utilized the S1K dataset (Muennighoff et al., 2025) and several models, each governed by specific licenses. The S1K dataset is released under the MIT License, which permits free use, modification, and distribution for any purpose, provided that the original copyright and license terms are retained. The Qwen2.5-7B and QwQ-Plus models (Yang et al., 2024) are licensed under Apache-2.0, allowing for both personal and commercial use, with the requirement to include copyright and license notices in any derivative works. Meanwhile, the DeepSeek r1 (Guo et al., 2025) and DeepSeek v3 (Liu et al., 2024) models are distributed under the MIT License, offering similar permissive terms for usage and distribution. These licensing frameworks ensure that our research complies with the conditions set forth by the respective developers and institutions, while also facilitating the transparent and lawful utilization of these artifacts in our study.

F AI Assistant Usage

During the research process, we utilized ChatGPT to polish some wording in the introduction section of the paper. We used the auto-completion feature of GitHub Copilot to assist with the coding. We did not overuse AI assistants in the writing and coding process. All text and code generated by AI assistants have been thoroughly checked and verified by us to avoid potential ethical issues and program errors. The final content and methods presented in this paper, as well as the coding work, are the original work of the authors.