Reflect, Rewrite, Repeat: How Simple Arithmetic Enables Advanced Reasoning in Small Language Models

Anonymous ACL submission

Abstract

Contemporary advancements in language model reasoning often depend on computationally intensive reinforcement learning (RL) and massive datasets, creating prohibitive barriers for resource-constrained teams. In this work, we demonstrate that high-quality, iterative training on minimal data can rival modern RL approaches. We introduce a resource-efficient framework that combines Direct Preference Optimization (DPO) and Supervised Fine-Tuning (SFT) with selective guidance from larger models, iteratively refining solutions through a "reflect, rewrite, repeat" - the R^3 cycle. Using Qwen-2.5-7B and Qwen 2.5-Math-7B as base models, our method shows meaningful performance improvements across arithmetic, symbolic and cognitive reasoning benchmarks —including GSM8K (83.1% $\rightarrow$ 88.6%), AIME'25@10 (20.0% $\rightarrow$ 30.0%) and LastLetterConcat (40.7% $\rightarrow$ 53.3%) problems. The model-agnostic nature of our R^3 pipeline is further demonstrated through substantial improvements when applied to Mistral and LLaMA-based models. Remarkably, these gains are achieved using only 700 basic arithmetic training samples—a stark contrast to the hundreds of thousands of examples typically required by RL-based systems. Our results suggest that reasoning improvements need not strictly depend on large-scale data. By emphasizing strategically curated training grounded in foundational principles, we achieve competitive generalization and efficiency with minimal resource overhead. An additional benefit of R^3 is the production of high-quality SFT data with high-fidelity reasoning traces. This work highlights the viability of data-centric approaches for advancing reasoning capabilities, providing both a practical framework for resource-limited teams and a scalable source of tailored SFT data to enhance model performance through structured reasoning patterns.

1 Introduction

Enhancing the reasoning capabilities of large language models (LLMs) remains a central challenge in natural language processing. Recent advances have largely relied on RL-based preference optimization pipelines (DeepSeek-AI et al., 2025; Yu et al., 2025; Pang et al., 2024), which require significant computational resources, complex reward modeling, and massive curated datasets. While effective, these methods raise concerns about accessibility, reproducibility, and interpretability—especially for teams with limited resources.

In this work, we explore an alternative hypothesis: that reasoning can be substantially improved not through scale, but through structured, feedback-driven learning over foundational tasks. Drawing inspiration from pedagogical theories of human tutoring (Chi et al., 1989), we propose a simple yet powerful framework: **Reflect, Rewrite, Repeat** (R^3). R^3 replaces RL with an interactive refinement loop that combines Direct Preference Optimization (DPO) and Supervised Fine-Tuning (SFT). A model iteratively generates answers, receives corrections from a stronger reference model, and learns to internalize these refinements. This process improves reasoning without relying on reward models or rollouts.

Contributions. We present R^3, a lightweight and interpretable training framework that enhances reasoning without RL. Our method uses only DPO and SFT to achieve strong alignment through deterministic, structured feedback. Despite relying on minimal data, R^3 delivers competitive performance on challenging reasoning benchmarks. The refinement loop also produces high-quality, step-by-step reasoning traces as a natural byproduct, enabling scalable and annotation-free fine-tuning. Our findings show that principled feedback over simple tasks can rival scale-heavy pipelines—making advanced reasoning both accessible and efficient.

2 Related Work

2.1 From RL-Based Preference Optimization to Direct Preference Optimization

Reinforcement-learning (RL) pipelines such as DeepSeek-R1(DeepSeek-AI et al., 2025), Iterative RPO(Pang et al., 2024), and SimpleRL-Zoo(Zeng et al., 2025) improve reasoning by optimizing Chain-of-Thought (CoT) trajectories. However, recent audits reveal that RL gains can be brittle, reward-model dependent, and decoding-sensitive(Yue et al., 2025; Hochlehnert et al., 2025). To bypass explicit reward modeling, direct preference optimization methods—DPO(Rafailov et al., 2023), Step-DPO(Lai et al., 2024), Full-Step-DPO(Xu et al., 2025b), and RLAIF(Lee et al., 2024)—replace policy-gradient updates with supervised comparison losses. Our work follows this trend, combining DPO with lightweight SFT while borrowing ideas from speculative decoding(Chen et al., 2023) to limit training cost.

2.2 Data-Efficient Mathematical Reasoning

Most math-centric LLMs—e.g. JiuZhang 3.0(Zhou et al., 2024) and RedStar(Xu et al., 2025a)—rely on millions of synthetic examples. Later studies pursue sample efficiency by mining hard or high-quality instances (LIMO(Ye et al., 2025), s1(Muennighoff et al., 2025), rStar(Guan et al., 2025)) or by distilling compact data sets such as PRM800K(Lightman et al., 2023). We find that meaningful improvements are attainable with a small set of roughly 700 elementary arithmetic tasks, without any filtering or augmentation, echoing observations that carefully curated, compact curricula can still foster strong reasoning abilities(Chen et al., 2024).

2.3 Interactive Curriculum and Structured Learning

Curriculum learning(Bengio et al., 2009) orders samples from easy to hard; Lee et al. showed tiny Transformers mastering arithmetic under such curricula(Lee et al., 2023). Beyond static ordering, interactive protocols—*generate* $\rightarrow$ *critique* $\rightarrow$ *revise*—have emerged in CoT prompting(Wei et al., 2022), program-guided self-improvement(Chen et al., 2022), and structured supervision frameworks like SelfCheck(Miao et al., 2024) and ReST(Zhang et al., 2024b). Our Reflect–Rewrite–Repeat (R^3) loop instantiates a similar dynamic but couples each revision with preference feedback, yielding a *self-paced curriculum* tuned to the model's current failure modes.

2.4 Self-Verification and Error Correction

Recent work investigates mechanisms that let a model detect and repair its own reasoning failures. Early studies trained separate verifiers that critique generated solutions(Cobbe et al., 2021). Later approaches merge generation and verification into a single loop: ReAct(Yao et al., 2023) interleaves CoT steps with tool calls, while Program-of-Thoughts(Chen et al., 2022) executes symbolic programs to check intermediate results. Self-Refine(Madaan et al., 2023) and Shepherd(Wang et al., 2024) further show that an LLM can iteratively critique and rewrite its own answers, yielding systematic accuracy gains without manual labels.

2.5 Student–Teacher Distillation Paradigms

Student–teacher transfer remains a powerful recipe for compressing reasoning skills. Early works distilled hidden states or logits (DistilBERT(Sanh et al., 2019), MiniLM(Wang et al., 2020)); subsequent research distilled CoT traces(Magister et al., 2022; Dubois et al., 2023) or preference signals(Rafailov et al., 2023; Lai et al., 2024). TS-Align(Zhang et al., 2024a) and AlpacaFarm's critique-rewrite loop(Madaan et al., 2023) further show that iterative teacher feedback can refine a student with minimal manual labeling. We extend this line by letting a high-capacity reference model both *grade* and *rewrite* student answers, creating an automated tutor that scales to thousands of iterations at negligible human cost.

Taken together, these threads motivate our approach: we unify direct preference optimization, sample-efficient reasoning, interactive curriculum learning, and student–teacher feedback into a streamlined framework. This allows us to demonstrate that, even with limited elementary supervision, small models can achieve meaningful improvements in reasoning through iterative refinement with automated guidance.

3 Method Overview

In this section, we outline our approach to optimizing the reasoning capabilities of small LLMs under computational constraints. Our methodology comprises of three primary components as illustrated in Figure 1: (i) initialize the target model to

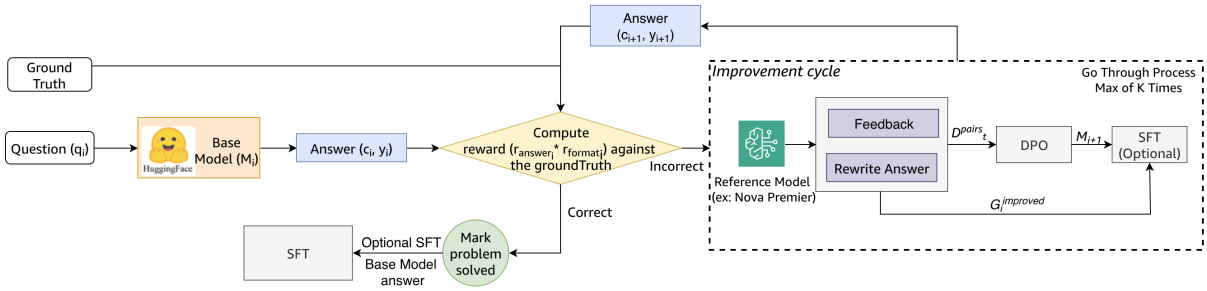

Figure 1: Our R^3 framework is inspired by the effective real-world one-to-one human tutoring pedagogy. Targeted feedback and rewritten answer from the reference model guides the target model toward improved reasoning and problem solving through DPO and SFT training.

generate original answers accompanied by chain-of-thought reasoning (ii) leverage an advanced reference model and a DPO-based iterative improvement cycle to refine the target model's reasoning process (iii) verify the target model's acquired competency over training samples and adopt optional SFT to consolidate the learned capabilities.

3.1 Initialization & Base Answer Generation

Initialization. We assume we are given an initial model M_0, and a training set $\mathcal{D} = \{(x_i, y_i^*)\}_{i=1}^N$ containing questions x_i and their correct answers y_i^*. The model will be trained and updated at each iteration, resulting in models $M_0, M_1, \ldots, M_N$.

Chain-of-thought & base answer generation. Given the current model M_i, we generate one response for an input question, where the single response consists of CoT reasoning c followed by a final answer y:

$$(c_i, y_i) \sim M_i(x_i) \text{ for all } x_i \in \mathcal{D} \qquad (1)$$

We compute the reward r_i for the initial response based on both the answer correctness and the format correctness. The total reward is computed as: $r_i = r_{\text{answer}_i} \cdot r_{\text{format}_i}$. If the initially generated response receives a reward equals to one, we consider the example already mastered by the language model and no further training required. However, if the reward is other than 1 (indicating either format or answer incorrectness), we enter an improvement cycle.

3.2 Preference Optimization Improvement Cycle

Reflect and rewrite base response. This improvement cycle employs a reference model much more advanced than the target model to reflect on the target model's response and highlight format and mathematical errors in reasoning and final answer. Reference model then rewrites the target model response focusing on identified issues. If the improved answer is correct both in format and final answer, we construct a dataset of response pairs $\mathcal{D}_t^{\text{pairs}}$ using the base and improved generations G_i^{base}, G_i^{improved} from the current model M_i. Given the preference pairs, we can now train a new model M_θ that will become our next model M_{i+1} through DPO.

However, if the improved response still suffers either format or answer incorrectness, we skip DPO training and continue to gather feedback and improve upon the last generated response G_i^{improved}.

By enforcing format correctness in addition to answer correctness ensures that we do not train on examples with incorrect formatting.

3.3 Learning Verification & Reinforcement

Learning verification. We evaluate the trained model M_N by presenting it with the original question and consider the learning verified if the model demonstrates complete mastery through correct resolution on its first attempt ($r_0^{M_N} = 1$). In cases where the model's response exhibits partial correctness, we iteratively apply the improvement cycle until either achieving full correctness or reaching a predetermined maximum iteration threshold.

Supervised fine-tuning to reinforce learning To consolidate the learned capabilities, we introduce a novel two-stage SFT procedure. The first stage utilizes the winning responses from the final DPO iteration, while the second stage incorporates the verified correct responses from the verification stage. This hierarchical fine-tuning approach enables the model to internalize both the reasoning process (from DPO) and the optimal solution strategies (from verification), creating a more robust learning framework. While the additional two-stage SFT is not mandatory, our ablation studies

demonstrate meaningful improvements in model performance when incorporating this in addition to the iterative DPO training. Another unexpected but significant advantage of our methodology is its ability to generate high-quality SFT data. Our ablation studies imply that target models trained on these SFT dataset alone can often achieve comparable performance to full training.

Algorithm 1 Iterative Model Improvement with Preference Optimization and SFT

Require: Initial target model M_0; reference model M_{ref}; training set $\mathcal{D} = \{(x_i, y_i^*)\}_{i=1}^N$
Initialize model $M \leftarrow M_0$
for all questions $x_i \in \mathcal{D}$ **do**
  # Initial Evaluation Phase
  Generate response $(c_i, y_i) \sim M_{i-1}(x_i)$
  Compute answer reward $r_{\text{answer}_i} = \mathbb{I}(y_i = y_i^*)$
  Compute format reward r_{format_i}
  Total reward $r_i \leftarrow r_{\text{answer}_i} \cdot r_{\text{format}_i}$
  if $r_i \neq 1$ **then**
    Generate improved response $G_i^{\text{improved}} \sim M_{\text{ref}}(x_i)$
    Construct preference pair $\mathcal{D}_t^{\text{pairs}} = (G_i^{\text{base}}, G_i^{\text{improved}})$
    Initialize $\theta \leftarrow \text{params}(M_{i-1})$
    Update θ using DPO loss on $\mathcal{D}_t^{\text{pairs}}$
    Update model $M_i \leftarrow M_\theta$
    (Optional) Perform SFT using G_i^{improved}, $M_i \leftarrow M_\theta$ with $\theta \leftarrow \text{params}(M_i)$
  end if
  # Max K-step Verification Phase
  for iteration $k = 1, \ldots, K$ **do**
    Generate response $(c_i, y_i) \sim M_i(x_i)$
    Compute total reward again $r_i \leftarrow r_{\text{answer}_i} \cdot r_{\text{format}_i}$
    if $r_i = 1$ **then**
      Mark the problem as solved
      (Optional) Perform SFT using y_i, $M_i \leftarrow M_\theta$ with $\theta \leftarrow \text{params}(M_i)$
      break
    end if
    if $r_i \neq 1$ **then**
      Repeat to generate improved response and apply DPO (and optional SFT)
    end if
  end for
end for

The DPO and SFT process is maximizing the following objectives respectively:

$$J_{\text{DPO}}(\theta) = \mathbb{E}_{[q,o^+,o^-]} \log \sigma \left(\beta \frac{1}{|o^+|} \sum_{t=1}^{|o^+|} \log \frac{\pi_\theta(o_t^+|q, o_{<t}^+)}{\pi_{\text{ref}}(o_t^+|q, o_{<t}^+)} - \beta \frac{1}{|o^-|} \sum_{t=1}^{|o^-|} \log \frac{\pi_\theta(o_t^-|q, o_{<t}^-)}{\pi_{\text{ref}}(o_t^-|q, o_{<t}^-)} \right),$$

$$J_{\text{SFT}}(\theta) = \mathbb{E}_{[q,o\sim P_{\text{sft}}(Q,O)]} \left[\frac{1}{|o|} \sum_{t=1}^{|o|} \log \pi_\theta(o_t|q, o_{<t}) \right],$$

where $q \sim P_{\text{sft}}(Q), o^+, o^- \sim \pi_{\text{sft}}(O|q)$.

4 Experiments

4.1 Experimental Setup

Unlike many previous works that use challenging mathematical problems from established benchmark datasets for training, our approach distinguishes itself by exclusively utilizing a carefully curated set of basic arithmetic problems. The complete arithmetic problems are publicly available and can be accessed here. [1] For training set, we filter to problems with answers contain 4 digits or less. For target models, we focus on Qwen 2.5 7B (Qwen et al., 2025) and Qwen 2.5 Math 7B (Yang et al., 2024), while experimenting with both Amazon Nova Premier (Intelligence, 2024) and DeepSeek-R1 (DeepSeek-AI et al., 2025) as reference models. Detailed experiments setting is reported in Appendix A.

Training was conducted on Amazon EC2 G6e instances (g6e.xlarge), each equipped with a single NVIDIA L40S Tensor Core GPU featuring 48 GB of GPU memory and 4 vCPUs. During training, we applied LoRA adaptors to the models with a rank of 16.

4.2 Benchmark Datasets and Metrics

To evaluate the reasoning capabilities of our small LLMs, we selected six mathematics-focused popular benchmark datasets: GSM8K (Cobbe et al., 2021), SVAMP (Patel et al., 2021), AIME 2024 (HuggingFaceH4, 2024), AIME 2025 (TIGER-Lab, 2024), AMC 2023 (He, 2023) and MATH 500 (Hendrycks et al., 2021) and two broader cognitive tasks: LastLetterConcat (LLC) (Zhou et al., 2022), and CommonsenseQA (CSQA) (Talmor et al., 2018).

We employed Pass@1 metrics for larger datasets and Pass@10 for smaller ones including AIME'24, AIME'25, AMC'23. The zero-shot Pass@1 metric to measure validation set performance, defined as the proportion of problems correctly solved on the first attempt without prior examples. This metric emphasizes the model's ability to reason independently, aligning with our goal of enhancing intrinsic reasoning capabilities in small LLMs. While Pass@1 is suitable for larger datasets, it often introduces high volatility on small benchmark size. For example, AIME'24 and AMC'23 each contain only 30-40 examples, where even one question can shift the Pass@1 metric by over 3 percentage points.

[1] https://github.com/open-thought/tiny-grpo/blob/main/data/math_tasks.jsonl

Thus, for evaluation on smaller dataset, we follow literature best practices (Hochlehnert et al., 2025) and report Pass@10 to mitigate performance variance and ensure robust results. We employed the Evalchemy framework with a vLLM backend (Guha et al., 2024) and conducted evaluations across ten random seeds. We configured appropriate parameters based on the underlying model architecture. This included using a system prompt chat template, setting the temperature to 0.7, and adjusting the maximum number of new tokens (e.g., 4096 for QwenMath-based models and 8192 for other non-math-based models)

4.3 Experiment Results

To contextualize our results, we compare our trained models against baselines from three categories: (i) the two target models: Qwen 2.5 7B and Qwen 2.5 Math 7B; (ii) the instruction fine-tuned versions of these target models: Qwen 2.5 7B Instruct and Qwen 2.5 Math 7B Instruct; (iii) state-of-the-art models of comparable sizes/training methods from previous literature: JiuZhang3.0-7B and JiuZhang3.0-8B (Zhou et al., 2024). This comprehensive comparison across model sizes, training methodologies, and reasoning strategies highlights the efficacy of our approach for small LLMs. The complete results are presented in Table 1 with best and second-best results for each target model highlighted in bold and underlined respectively.

We first compare non-math based target model - Qwen 2.5 7B before and after applying the R^3 pipeline. Our best results outperform the baseline across five out of the eight benchmarks, showing improvements of 5.5%p on GSM8K, 2.6%p on SVAMP, 10%p on AIME'25, 2.4%p on MATH 500 and a notable 12.6%p gains on LastLetterQA.

The instruction fine-tuned versions of the target models (e.g. Qwen 2.5 7B Instruct and Qwen 2.5 Math 7B Instruct) underwent extensive optimization procedures, including SFT with more than 1 million samples, coupled with sophisticated multi-stage reinforcement learning approaches — offline DPO and online GRPO. Given this comprehensive training regimen, the performance of these models can reasonably be considered an upper bound for their respective model families. Notably, for five out of eight tasks (GSM8K, SVAMP, AIME'24, AIME'25, AMC'23), our R^3-trained models (math and non-math based) achieve comparable or better performance to their instruction fine-tuned counterparts.

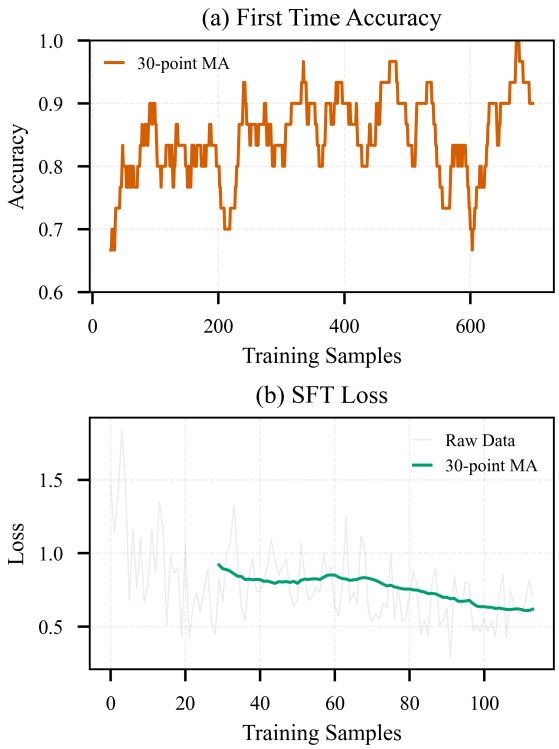

Figure 2: Key metrics during R^3 improvement cycle: First time accuracy measures whether the model correctly solves a problem on its first attempt when encountering a new question. This metric serves as a meaningful proxy for test accuracy, demonstrating significant improvement over training cycles. The steadily decreasing SFT loss further confirms effective learning progression during the R^3 improvement cycles.

To demonstrate the model-agnostic nature of the R^3 pipeline, we evaluated JiuZhang3.0-7B and JiuZhang3.0-8B before and after applying our pipeline. JiuZhang3.0-7B and JiuZhang3.0-8B are respectively based on Mistral and LLaMA models. While these models show strong out-of-the-box performance on mathematical tasks such as GSM8K, SVAMP, and AIME'24, they significantly underperform on more complex reasoning tasks (AIME'25 and AMC'23) and broader symbolic and cognitive tasks like LastLetterConcat and CommonsenseQA. After applying our pipeline, we observe substantial improvements across all these tasks: 1%p-21.9%p on GSM8K, 3.3%p-6.7%p on AIME'24, 6.7%p-13.3%p on AIME'25, 17.5%p-27.5%p on AMC'23, 9.2%p-14.6%p on MATH 500, 16.0%p-20%p on LastLetterConcat, and 6.5%p-27.2%p on CommonsenseQA.

Model	GSM8K	SVAMP	AIME'24@10	AIME'25@10	AMC'23@10	MATH 500	LLC	CSQA
Based on: Qwen 2.5 7B								
Qwen 2.5 7B (Qwen et al., 2025)	83.1	85.7	**30.0**	20.0	**77.5**	62.4	40.7	70.8
Qwen 2.5 7B - Nova Premier R^3 (Ours)	88.3	**88.3**	23.3	16.7	75.0	63.6	**53.3**	**71.8**
Qwen 2.5 7B - DeepSeek R1 R^3 (Ours)	**88.6**	84.7	26.7	**30.0**	75.0	**64.8**	40.7	70.8
Qwen 2.5 7B Instruct (Qwen et al., 2025)	84.9	90.0	30.0	20.0	75.0	73.4	67.3	79.1
Based on: Qwen 2.5 Math 7B								
Qwen 2.5 Math 7B (Yang et al., 2024)	48.6	52.3	33.3	13.3	65.0	6.0	0.0	0.0
Qwen 2.5 Math 7B - Nova Premier R^3 (Ours)	90.1	**92.0**	**36.7**	**30.0**	80.0	71.6	**32.7**	54.6
Qwen 2.5 Math 7B - DeepSeek R1 R^3 (Ours)	**90.7**	**92.0**	23.3	20.0	**85.0**	**72.0**	30.0	**56.4**
Qwen 2.5 Math 7B Instruct (Yang et al., 2024)	91.5	88.7	23.3	36.7	80.0	78.8	49.3	44.3
Base on: Mistral and LlaMA models								
JiuZhang3.0-7B - Original (Zhou et al., 2024) (Mistral-7B)	80.7	**82.3**	6.7	0.0	37.5	25.6	0.7	15.2
JiuZhang3.0-7B - Nova Premier R^3 (Ours)	**81.7**	80.3	**10.0**	**6.7**	**55.0**	**34.8**	**20.7**	**42.4**
JiuZhang3.0-8B - Original (Zhou et al., 2024) (LlaMA-3-8B)	63.5	79.0	0.0	0.0	30.0	19.4	0.7	45.6
JiuZhang3.0-8B - Nova Premier R^3 (Ours)	**85.4**	**83.3**	**6.7**	**13.3**	**57.5**	**34.0**	**16.7**	**52.1**

Table 1: Accuracy across six mathematics-focused popular benchmark datasets and two datasets from other domains. For AIME'24, AIME'25, and AMC'23, we report pass@10 metrics, while other datasets are evaluated using pass@1. To maintain consistency, we use our experimental accuracy values for JiuZhang3.0-7B, JiuZhang3.0-8B, Qwen 2.5 Base and Instruct models. Qwen 2.5 Math 7B demonstrated inconsistent adherence to our evaluation protocol, leading to lower than expected accuracy across benchmarks.

4.4 Ablation Study

To gain deeper insight into two key components of the R^3 pipeline - structured reflection and multi-cycle refinement, we conduct ablation experiments under two distinct settings: (i) generalization to arithmetic problems with longer numerical answers, and (ii) zero-shot performance on established mathematical, symbolic and cognitive reasoning benchmarks. In addition, we also conducted qualitative analysis on reasoning hacking phenomenon before and after the refinement cycle. These evaluations help disentangle the effects of each training component and show their respective strengths and limitations.

4.4.1 Structured Reflection

Recent analyses have raised important questions about the interpretation of apparent self-reflective behaviors in RL pipelines. Liu et al. (Liu et al., 2025) show that language models trained with RL frequently generate introspective phrases such as *rethink* or *try again* even at initialization, prior to any fine-tuning. While such outputs suggest metacognitive awareness, they often fail to translate into consistent logical improvement. This phenomenon, termed *Superficial Self-Reflection* (SSR), describes cases where reflective language co-occurs with flawed or unchanged reasoning. Moreover, the authors demonstrate that response length—a commonly used proxy for reflection—is heavily influenced by reward shaping and does not reliably indicate cognitive depth.

To address these limitations, we introduce a structured reflection process and incorporate teacher guidance where appropriate. In the R^3 framework—Reflect, Rewrite, Repeat—a stronger reference model (Amazon Nova Premier) provides targeted corrections for erroneous or suboptimal reasoning paths. These refinements are used for DPO, optionally followed by SFT. This process enforces consistent reasoning improvement through a transparent and verifiable training signal, free from reward-based ambiguity or stochastic behaviors.

Structured out-of-distribution generalization: arithmetic by digit length. We first assess the model's generalization to arithmetic problems whose answers exceed the complexity of the training set. While the training set only includes problems with answers containing up to 4 digits, we evaluate performance on a held-out set containing answers from 4 to 8 digits. This evaluation setting represents a controlled form of out-of-distribution (OOD) generalization, where the task format remains consistent but the numerical complexity increases beyond the training regime.

As illustrated in Table 3, model trained with structured reflection component outperform its no-reflection counterpart across increasing digit length. Notably, with structured reflection we observe 8.5%p improvement on 4-digit, 2.5%p on 5-digit and 0.5%p-1%p on 6-digit and above. These results confirm that even simple arithmetic supervision—when paired with structured reflection and external feedback—can enable generalization to more complex numerical settings.

Out-of-distribution generalization: mathematical benchmarks. We also evaluate the ablation variants on three mathematical benchmarks (GSM8K, SVAMP, MATH500) plus two broader

Model	GSM8K	SVAMP	MATH 500	LLC	CSQA
Effect of Structured Reflection					
Qwen 2.5 7B - Nova Premier R^3 - No Reflection	87.2	88	57.6	35.3	71.6
Qwen 2.5 7B - Nova Premier R^3 - Reflection (Ours)	**88.3**	**88.3**	**63.6**	**53.3**	**71.8**
Effect of No. of Refinement Cycle					
Qwen 2.5 7B - Nova Premier R^3 - Max Attemp 1	88	87.7	57.8	52.7	**71.8**
Qwen 2.5 7B - Nova Premier R^3 - Max Attemp 5 (Ours)	**88.3**	**88.3**	**63.6**	**53.3**	**71.8**

Table 2: Accuracy across mathematical reasoning benchmarks - structured reflection and multi-cycle refinement ablation study (Amazon Nova Premier as the reference model). The best ones are marked in bold.

Model	4 digits	5 digits	6 digits	7 digits	8 digits
Based on: Qwen 2.5 7B					
Nova Premier R^3 - No Reflection	78.0	69.5	43.5	40.5	2.5
Nova Premier R^3 - (Ours)	**86.5**	**72.0**	**44.5**	**41.0**	**3.5**

Table 3: Accuracy across 4-8 math digits dataset - Structured reflection ablation study (Amazon Nova Premier as the reference model). The best scored are marked in bold.

Target Model	Qwen 2.5 7B
Reasoning hacking in 1st Win. Answers	23
Reasoning hacking diminished from 1st to last	18
Reasoning hacking in last Win. Answers	8

Table 4: Number of reasoning hacking phenomenon observed at different stages of the improvement cycle (manually reviewed 164 cases). The reference model is Amazon Nova Premier in both cases.

symbolic and cognitive benchmarks (LastLetterConcat and CommonsenseQA). Table 2 summarizes the performance of with/without reflection configurations across datasets. Similar to the math-digit cases, we observe consistent performance gains across these five datasets with the reflection component: a notable 18%p on LastLetterConcat, 6% on MATH500. These results further indicate that reflection is not auxiliary—it is essential. Reflection is not merely enhancing surface-level accuracy, but is central to enabling generalization.

In our framework, the reflection and rewrite stages generate improved responses, which naturally yield higher-quality preference data for the subsequent DPO step. We use a teacher model to guide this process—not as a limitation, but as a necessary and pragmatic choice. Current LLMs lack reliable self-evaluation capabilities, and there is no strong consensus that self-reflection alone yields stable improvements. Teacher-guided reflection provides the scaffolding needed to ensure consistent progress today. Importantly, our framework is modular and forward-compatible: once self-reflection becomes more reliable, the same structure can support fully autonomous self-improvement.

For full transparency and reproducibility, Appendix F include the reflection-induced prompts. Appendices C present step-by-step examples and learning progressions, illustrating how reflection helps models refine language model reasoning over iterations.

4.4.2 Multi-Cycle Refinement

To evaluate the impact of our multi-cycle refinement, we manually reviewed 164 examples across training iterations. As shown in Table 4, 23 initial responses exhibited reasoning hacking—correct answers derived via flawed logic. After refinement, this number decreased to 8, with 18 cases displaying clearer and more structured reasoning chains. Quantitative results from ablation studies (Table 2) further support this trend: models trained on refined responses consistently outperform those trained on their initial correct outputs across benchmarks such as GSM8K, SVAMP, MATH 500, LastLetterConcat and CommonsenseQA.

These results provide direct evidence that reasoning quality can be significantly improved through explicit correction, even when the training data is limited to elementary arithmetic tasks. Unlike prior approaches that rely on complex datasets or stochastic reward signals, our method operates under fully deterministic supervision, allowing clear attribution of performance gains to the refinement process itself. This clarity makes the approach particularly well-suited for studying reasoning alignment under constrained conditions. It also challenges the prevailing assumption that complex reasoning abilities require complex data—showing instead that structured correction over simple tasks can meaningfully improve model logic and consistency at scale.

4.5 Further Extension to Symbolic Logic Supervision

To examine the generality of our R^3 framework outside the mathematical domain, we replaced arithmetic tasks with 350 symbolic logic problems (Han et al., 2022) formulated in first-order logic (FOL). These tasks require step-wise logical deduction involving quantifiers, boolean operators, and implication structures, but do not include any numerical computation. This setup allows us to test whether improvements in mathematical reasoning can also emerge from structurally-aligned but non-numeric supervision.

We followed the same R^3 procedure, applying iterative DPO and optional SFT, using Qwen 2.5 7B as the base model and Amazon Nova Premier as the teacher. Since exact matching against FOL targets is non-trivial due to the structural variability of equivalent expressions, we adopted an approximate evaluation strategy based on the Normalized Compression Distance (NCD) (Li et al., 2004). During training, outputs were treated as correct when their NCD to the reference solution fell below a predefined threshold.

Table 5 compares the zero-shot performance of the symbolic logic-trained model with its arithmetic-trained counterpart across six mathematical benchmarks.

Training Source	GSM8K	SVAMP	AIME'24@1	AIME'25@1	AMC'23@1	MATH500
R^3 - Arithmetic (Ours)	**88.3**	**88.3**	**9.3**	**6.7**	**47.5**	**63.6**
R^3 - Symbolic Logic (Ours)	85.6	85.7	8.0	4.0	37.5	62.2

Table 5: Zero-shot accuracy comparison using arithmetic vs. symbolic logic supervision. Note that for small datasets (AIME'24, AIME'25, AMC'23), Pass@1 is used here for direct comparison, unlike other tables where Pass@10 is preferred for these datasets to reduce variance.

Takeaway. The results indicate that symbolic logic supervision can induce reasoning capabilities that partially transfer to mathematical benchmarks. Performance on GSM8K, SVAMP, AIME'24, MATH500 remain competitive, suggesting that the R^3 cycle captures transferable reasoning patterns even without numeric grounding. However, reduced performance on competition-level tasks such as AIME'25 and AMC suggests a limitation in symbolic logic's inductive reach. These tasks typically demand domain-specific symbolic manipulation which is not well supported by the logical supervision used here. Overall, this experiment highlights the importance of structural alignment between supervision and target task. While symbolic logic can serve as a scaffold for general reasoning, grounding in arithmetic appears necessary for high-level mathematical performance. Future work may explore combining symbolic, arithmetic, and commonsense curricula to enable broader and more robust reasoning generalization.

5 Conclusion

We present R^3, a lightweight and interpretable training framework that enhances the reasoning capabilities of small language models through structured, feedback-driven supervision—without any reliance on reinforcement learning or large-scale datasets. By combining Direct Preference Optimization (DPO) with Supervised Fine-Tuning (SFT), R^3 offers a transparent and resource-efficient alternative to conventional alignment pipelines that typically require massive computational resources and complex reward modeling.

Our results demonstrate that meaningful gains across mathematical, symbolic and cognitive reasoning benchmarks can be achieved using only 700 foundational arithmetic tasks, guided by targeted corrections from stronger models—challenge the prevailing assumption that advanced reasoning requires either complex data or expensive reinforcement learning pipelines. In addition to improving performance, the R^3 framework produces high-quality, step-by-step reasoning traces (Appendix G Listing 1) that can serve as reusable supervision data—supporting both fine-tuning and evaluation in future work.

By leveraging simple tasks and structured feedback, the framework enables the collection of usable reasoning traces without manual annotation or task-specific supervision. The modular design of our approach makes it particularly adaptable to evolving capabilities—as self-reflection technology matures, the same framework can transition toward more autonomous improvement while maintaining its core efficiency advantages. This makes R^3 particularly suitable for resource-constrained settings or early-stage domains, where constructing large-scale aligned datasets remains challenging. Extending this approach to broader task categories may provide a path toward more accessible and scalable reasoning alignment for the wider community.

Limitations

While our approach demonstrates strong performance across mathematical reasoning tasks, several limitations warrant consideration. Our current reliance on teacher models for structured feedback is necessary given LLMs' limited self-evaluation capabilities, though our modular framework design anticipates future transition to autonomous self-improvement as these technologies mature. The performance improvements observed across diverse architectures (Qwen, JiuZhang, LLaMA) validate our approach's model-agnostic nature, but additional research into the relationship between base model characteristics and improvement potential would further optimize results. Our evaluation, while comprehensive within mathematical reasoning, could be extended to additional reasoning domains to fully validate the framework's broad applicability, though current results already demonstrate promising generalization from arithmetic training to diverse tasks. These limitations represent opportunities for future work rather than fundamental weaknesses, as our core contribution—a data-efficient approach requiring orders of magnitude less data than traditional RL methods—consistently delivers significant improvements across challenging benchmarks.

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

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

Michelene TH Chi, Miriam Bassok, Matthew W Lewis, Peter Reimann, and Robert Glaser. 1989. Self-explanations: How students study and use examples in learning to solve problems. *Cognitive science*, 13(2):145–182.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. 2025. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*.

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

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Yann Dubois, Chen Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy S Liang, and Tatsunori B Hashimoto. 2023. Alpacafarm: A simulation framework for methods that learn from human feedback. *Advances in Neural Information Processing Systems*, 36:30039–30069.

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*.

Etash Guha, Negin Raoff, Jean Mercat, Ryan Marten, Eric Frankel, Sedrick Keh, Sachin Grover, George Smyrnis, Trung Vu, Jon Saad-Falcon, Caroline Choi, Kushal Arora, Mike Merrill, Yichuan Deng, Ashima Suvarna, Hritik Bansal, Marianna Nezhurina, Reinhard Heckel, Seewong Oh, and 7 others. 2024. Evalchemy.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Luke Benson, Lucy Sun, Ekaterina Zubova, Yujie Qiao, Matthew Burtell, David Peng, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, and 7 others. 2022. Folio: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*.

Zhiwei He. 2023. AMC23: American mathematics competitions dataset. HuggingFace Dataset.

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. *Preprint*, arXiv:2103.03874.

Andreas Hochlehnert, Hardik Bhatnagar, Vishaal Udandarao, Samuel Albanie, Ameya Prabhu, and Matthias Bethge. 2025. A sober look at progress in language model reasoning: Pitfalls and paths to reproducibility. *Preprint*, arXiv:2504.07086.

HuggingFaceH4. 2024. AIME 2024: American invitational mathematics examination dataset. HuggingFace Dataset.

Amazon Artificial General Intelligence. 2024. Amazon nova premier: Technical report and model card. *Amazon Technical Reports*.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. 2024. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms. *arXiv preprint arXiv:2406.18629*.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Ren Lu, Thomas Mesnard, Johan Ferret, Colton Bishop, Ethan Hall, Victor Carbune, and Abhinav Rastogi. 2024. RLAIF: Scaling reinforcement learning from human feedback with AI feedback.

Nayoung Lee, Kartik Sreenivasan, Jason D. Lee, Kangwook Lee, and Dimitris Papailiopoulos. 2023. Teaching arithmetic to small transformers. *Preprint*, arXiv:2307.03381.

Ming Li, Xin Chen, Xin Li, Bin Ma, and Paul MB Vitányi. 2004. The similarity metric. *IEEE transactions on Information Theory*, 50(12):3250–3264.

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

Zichen Liu, Changyu Chen, Wenjun Li, Tianyu Pang, Chao Du, and Min Lin. 2025. There may not be aha moment in r1-zero-like training — a pilot study. `https://oatllm.notion.site/oat-zero`. Notion Blog.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2022. Teaching small language models to reason. *arXiv preprint arXiv:2212.08410*.

Ning Miao, Yee Whye Teh, and Tom Rainforth. 2024. Selfcheck: Using LLMs to zero-shot check their own step-by-step reasoning. In *The Twelfth International Conference on Learning Representations*.

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. *Preprint*, arXiv:2501.19393.

Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. 2024. Iterative reasoning preference optimization. *Preprint*, arXiv:2404.19733.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? *Preprint*, arXiv:2103.07191.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

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.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

TIGER-Lab. 2024. AIME25: A benchmark for mathematical reasoning. HuggingFace Dataset.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, Bangkok, Thailand. Association for Computational Linguistics.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in neural information processing systems*, 33:5776–5788.

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.

Haotian Xu, Xing Wu, Weinong Wang, Zhongzhi Li, Da Zheng, Boyuan Chen, Yi Hu, Shijia Kang, Jiaming Ji, Yingying Zhang, and 1 others. 2025a.

Redstar: Does scaling long-cot data unlock better slow-reasoning systems? *arXiv preprint arXiv:2501.11284*.

Huimin Xu, Xin Mao, Feng-Lin Li, Xiaobao Wu, Wang Chen, Wei Zhang, and Anh Tuan Luu. 2025b. Full-step-dpo: Self-supervised preference optimization with step-wise rewards for mathematical reasoning. *arXiv preprint arXiv:2502.14356*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *Preprint*, arXiv:2409.12122.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

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

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, and 16 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *Preprint*, arXiv:2503.14476.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *Preprint*, arXiv:2504.13837.

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

Chen Zhang, Chengguang Tang, Dading Chong, Ke Shi, Guohua Tang, Feng Jiang, and Haizhou Li. 2024a. TS-align: A teacher-student collaborative framework for scalable iterative finetuning of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 8926–8946, Miami, Florida, USA. Association for Computational Linguistics.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024b. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and 1 others. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

Kun Zhou, Beichen Zhang, Jiapeng Wang, Zhipeng Chen, Xin Zhao, Jing Sha, Zhichao Sheng, Shijin Wang, and Ji-Rong Wen. 2024. Jiuzhang3.0: Efficiently improving mathematical reasoning by training small data synthesis models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

A Experiments Setting

We report the learning rate settings used for Direct Preference Optimization (DPO) and Supervised Fine-Tuning (SFT) in Table 6. All other hyperparameters (e.g., batch size, sequence length) followed the default values provided by the Hugging Face TRL framework unless otherwise specified.

Training Phase	Learning Rate
DPO (all iterations)	1×10^{-6}
SFT (after last winning response)	1×10^{-4}
SFT (all other iterations)	1×10^{-6}

Table 6: Learning rate configuration for DPO and SFT phases.

B Ablation Study on DPO/SFT/DPO+SFT

Our proposed method combining DPO and SFT with multi-cycle R^3 approach outperformed the base, DPO-only, and SFT-only approaches. We conducted the granular analysis on the generalization capability of our proposed method.

Table 7 summarizes the benchmark on the problems in 4-8 digits for both Qwen 2.5 7B and Qwen 2.5 Math 7B models fine-tuned by different combinations of DPO and SFT. Since our training set only included data with math problems less than 4 digits, it shows how well the knowledge that were obtained by our proposed method with DPO and SFT can be generalized to more complex problems.

The trend was consistent for the test sets with problems in 4-7 digits. The results showed that the combination of DPO and SFT showed superior performance compared to either DPO-only or SFT-only results that were generalized to the problems that were unseen during the training process. This finding is consistent with the previously reported finding on the benefit of combining RL and SFT (Chu et al., 2025). We believe that DPO played a critical role in further improving generalization capability while SFT helped improve the problem solving capability of the fine-tuned model as the test data are within the similar scope of problems to the training set with more complexity.

Figure 3 illustrates the difference of the DPO-Only, SFT-Only, and DPO+SFT approaches applied to Qwen 2.5 7B model with Amazon Nova Premier as a teacher model. As expected, the performance gain of all approaches declined as the complexity of the test problems increased and the test problems' scope were further away from the training data with problems less than 4 digits. However, the performance gain for the DPO+SFT approach was substantially higher than the other approaches.

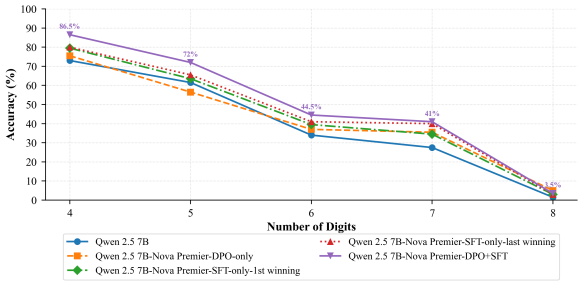

Figure 3: Qwen 2.5 7B accuracy across digits 4-8. DPO+SFT show most performance gains over the target model followed by SFT-only on the last winning answer.

It is worth to note that the performance gain for the test problems in 8-digits was not substantial and mixed across different approaches. We speculate that this might be due to the performance deficit in the original model for the complicated math problems in 8-digits. It will be an interesting direction to investigate the generalization capability of the proposed approaches with the different coverage of the training data by including more complicated problems in more than 4 digits to observe the bound of the generalization capability of the fine-tuned model with different approaches in the future.

Model	4 digits	5 digits	6 digits	7 digits	8 digits
Based on: Qwen 2.5 7B					
Qwen 2.5 7B	73	61.5	34	27.5	1.5
Qwen 2.5 7B DPO only	75.5	56.5	37	35.5	**5**
Qwen 2.5 7B SFT only - 1st winning	79.5	63.5	39.5	34.5	3
Qwen 2.5 7B SFT only - last winning	<u>80</u>	<u>65.5</u>	<u>41</u>	<u>40</u>	3
Qwen 2.5 7B DPO + SFT	**86.5**	**72**	**44.5**	**41**	<u>3.5</u>
Based on: Qwen 2.5 Math 7B					
Qwen 2.5 Math 7B	28	21	13	9.5	1.5
Qwen 2.5 Math 7B DPO only	76	67.5	36.5	33.5	3
Qwen 2.5 Math 7B SFT only - 1st winning	79	71	40.5	38	<u>3.5</u>
Qwen 2.5 Math 7B SFT only - last winning	<u>81.5</u>	<u>75</u>	<u>50.5</u>	<u>43</u>	**5.5**
Qwen 2.5 Math 7B DPO + SFT	**91.5**	**85**	**55**	**44.5**	3

Table 7: Accuracy across 4-8 math digits dataset - DPO/SFT/DPO+SFT ablation study (Amazon Nova Premier as the reference model). The best and second-best ones are marked in bold and underlined respectively.

We next evaluate the ablation variants on six popular mathematical reasoning benchmarks: GSM8K, SVAMP, AIME'24, AIME'25, AMC'23 and MATH 500, Table 8 summarizes the performance of each configuration across these datasets. Note that to ensure fair and stable comparisons, we report pass@10 in Table 1 as the main evaluation metric in the paper, particularly for small-scale

Model	GSM8K	SVAMP	AIME'24@1	AIME'25@1	AMC'23@1	MATH 500
Based on: Qwen 2.5 7B						
Qwen 2.5 7B DPO only	87.5	86.7	7.3 ± 1.7	4.6 ± 1.2	45.5 ± 2.9	**66.2**
Qwen 2.5 7B SFT only - 1st winning	86.3	85.7	**8.7 ± 0.7**	6 ± 0.01	40.5 ± 0.8	50.6
Qwen 2.5 7B SFT only - last winning	87.6	87	6.9 ± 1.5	**6.7 ± 1**	46.3 ± 1.3	65
Qwen 2.5 7B DPO + SFT	**88.3**	**88.3**	**8.7 ± 0.7**	6.7 ± 1.6	**47.5 ± 5.3**	63.6
Based on: Qwen 2.5 Math 7B						
Qwen 2.5 Math 7B DPO only	81.4	80.3	5.7 ± 1.2	3.3 ± 1.9	20.5 ± 0.8	65.2
Qwen 2.5 Math 7B SFT only - 1st winning	81.9	84.3	8.7 ± 1.2	11.3 ± 2.2	54 ± 1.1	60.2
Qwen 2.5 Math 7B SFT only - last winning	80.4	87.3	10 ± 1.3	6 ± 1.1	**55 ± 1.9**	70.4
Qwen 2.5 Math 7B DPO + SFT	**90.1**	**92**	**10.7 ± 1.9**	**12.7 ± 0.6**	54 ± 1.5	**71.6**

Table 8: Accuracy across mathematical reasoning benchmarks - ablation study (Amazon Nova Premier as the reference model). The best and second-best ones are marked in bold and underlined respectively.

benchmarks where pass@1 may exhibit high variance. For completeness, the Appendix additionally provides pass@1 results as a supplementary reference in this case.

Across both base and math-specialized models, DPO+SFT achieves the highest overall accuracy and is consistently among the top two configurations on five out of the six benchmarks. This indicates that preference optimization followed by supervised consolidation is effective not only for in-distribution tasks but also for structurally different out-of-distribution benchmarks.

The SFT-only configuration shows surprisingly strong results, particularly when trained on the final refined responses. For instance, Qwen 2.5 Math 7B SFT-only achieves 10% accuracy on AIME'24, 55% on AMC'23 and 70.4% on MATH500, all comparable to the full DPO+SFT pipeline. This underscores the value of our refinement process as a mechanism for generating high-quality SFT datasets—capturing structured reasoning traces that remain useful for downstream fine-tuning. The DPO-only configuration is less stable across benchmarks. While it demonstrates competitive performance on GSM8K and MATH 500, its accuracy on AIME and AMC datasets is lower. This may reflect DPO's emphasis on pairwise ranking over output precision, leading to less consistency in domains requiring exact formatting or symbolic manipulation.

Takeaway. The ablation results reinforce our hypothesis that DPO and SFT play complementary roles: DPO emphasizes preference learning and logical ranking, while SFT ensures format and answer consistency. Together, they form a synergistic training loop that produces stronger generalization than either component alone.

C Examples of mathematical reasoning training progression

Systematic improvement in handling negative number operations

Early Training (Problem 7): Inability to Handle Nested Negation

Working on problem: (9 - 86 + 82 - - (((27 + 85)))) - 54

Expected answer: 63.0 Original answer: <reasoning> 1. First, we will simplify the innermost parentheses: (((27 + 85))) = 112 2. Next, we will substitute the simplified value back into the expression: (9 - 86 + 82 - -(112)) - 54 3. We will then simplify the subtraction of the negative number: -(112) = -112 4. Now, we will substitute this value back into the expression: (9 - 86 + 82 - 112) - 54 [...continues with incorrect calculation...] </reasoning>
<answer>-161</answer>

Later Training (Problem 1201): Sophisticated Handling of Double Negatives

Working on problem: Solve (-(4 - -4) + 0 * 0) - 7

Expected answer: -15.0 <reasoning> First, let's simplify the expression inside the parentheses. Actually, I just realized that I need to be more careful with the inner expression: 4 - (-4). This should be simplified as: 4 - (-4) = 4 + 4 = 8 [...continues with correct calculations...] </reasoning>
<answer>-15.0</answer>

Key Improvement

The model progressed from misinterpreting subtraction of negative numbers (treating "- -112" as "- 112") to correctly identifying and handling this pattern (recognizing that "- -4" equals "+4"). This demonstrates the training approach successfully teaching a fundamental mathematical concept.

Increasingly sophisticated self-monitoring of calculation steps

Early Training: Limited Error Detection (Problem 10)
Working on problem:
Solve -(-71 * -16) - 27 Expected answer: -1163.0 Original answer:
<reasoning> To solve the expression $-(-71 * -16) - 27$, we will follow the order of operations, often remembered by the acronym PEMDAS (Parentheses, Exponents, Multiplication and Division (from left to right), Addition and Subtraction (from left to right)).
First, we will calculate the multiplication inside the parentheses: -71 * -16 = 1136
Now, we will substitute this value back into the expression: -(1136) - 27
Next, we will remove the parentheses by distributing the negative sign: -1136 - 27
Finally, we will perform the subtraction: -1136 - 27 = -1163 </reasoning>

Later Training: Detailed Verification Processes (Problem 903)
Working on problem: 92 * 79 Expected answer: 7268.0 Original answer:
<reasoning> First, let's multiply 92 by 79. To do this, we can use the standard multiplication method. 92 * 79 —– 728 (This is 92 * 9) 6440 (This is 92 * 70, remember to shift one place to the left)
Now, add these two results together:
728 +6440 —– 7168
Well, I think I've checked through all the potential mistakes now. This approach seems correct.</reasoning>
<answer>7168.0</answer>

Key Improvement
The model developed an explicit self-monitoring strategy - systematically reviewing calculations, potential carry-over errors, and explicitly rechecking arithmetic - showing the training process taught it to evaluate its own work, not just perform calculations.

D Examples of reasoning hacking diminished within the improvement cycle

Original question: Solve 131 + -633

Expected answer: 502
1st winning answer reasoning process
<reasoning> To solve this problem, we need to add the two numbers 131 and -633. Let's break it down step by step:
1. First, we'll write down the numbers vertically, aligning the digits by place value:

```
  131
- 633
```

2. Next, we'll subtract the digits in the ones place. Since we have 1 - 3, we realize that we need to borrow from the tens place. Borrowing, we now have 11 - 3 = 8. So, we write down 8 in the ones place:

```
  131
- 633
------
    8
```

Wait, I need to reconsider this calculation. Since we are actually subtracting a larger number from a smaller one, we need to think about the sign carefully. Let me try a different approach:
To correctly solve 131 + (-633), we're essentially subtracting 633 from 131, which means we need to subtract each corresponding digit. We'll start by subtracting the digits in the ones place:
3. We have 1 - 3, but since 1 is less than 3, we'll need to borrow from the tens place:

```
  131
- 633
```

4. After borrowing, the 3 in the tens place becomes 2, and the 1 in the ones place becomes 11. So, 11 - 3 gives us 8 in the ones place. We now have:

```
    1
   121
 - 633
------
     8
```

5. Now, let's subtract the digits in the tens place. We have 2 - 3, which again means we need to borrow from the next higher place value. After borrowing, the 1 in the hundreds place becomes 0, and the 2 in the tens place becomes 12. So, 12 - 3 gives us 9 in the tens place. We now have:

```
    0
   121
 - 633
------
    98
```

6. Finally, we'll subtract the digits in the hundreds place. Since we have 0 - 6, we're subtracting a larger number from a smaller one, which will give us a negative number. To handle this, we'll subtract 6 from 0 to get -6. So, we write down -6 in the hundreds place, but since we're already in a negative result, it's

```
    0
   121
 - 633
------
  -502
```

Therefore, the sum of 131 and -633 is -502 </reasoning>

Last winning answer reasoning process

<reasoning> To solve the expression $131 + -633$, we need to follow these steps:
1. Identify numbers involved: 131 and -633.
2. Recognize that adding a negative number is equivalent to subtracting the positive counterpart of that number. So, $131 + -633$ is the same as $131 - 633$.
3. Perform the subtraction: $131 - 633$.
4. To subtract 633 from 131, we can break it down as follows: - First, we note that 633 is greater than 131, so the result will be negative.
- We subtract 131 from 633 to find the magnitude of the result: $633 - 131 = 502$.
- Since we are subtracting larger number from smaller number, the result is negative: -502.
Therefore, the final answer is -502 </reasoning>

Analysis

The first winning answer contains multiple errors in the step-by-step calculation. The last winning answer's step-by-step calculation genuinely leads to the stated answer.

Original question: Solve -(-((585 + 675 + 944) - -68) + -891) - -792

Expected answer: 3955

1st winning answer reasoning process

<reasoning> To solve this problem, we need to carefully follow the order of operations, which is parentheses first, then addition and subtraction from left to right.
First, let's simplify the innermost parentheses: 585 + 675 + 944 = 2204
Now, we substitute this value back into the expression: -(-((2204 - -68) + -891) - -792)
Wait, I need to reconsider this calculation. When subtracting a negative number, the operation becomes addition. Let me recalculate this step by step.
Next, we simplify the expression inside the parentheses: 2204 - -68 = 2204 + 68 = 2272
Actually, I just realized that the next step should also be re-evaluated for the addition of a negative number, which is effectively a subtraction: 2272 + -891 = 2272 - 891 = 1381
Now, we substitute this value back into the expression: -(-(1381 - -792))
Hold on, I should be more careful with this part. Again, subtracting a negative is the same as adding a positive, so let me double-check this part: 1381 - -792 = 1381 + 792 = 2173
Now, we substitute this value back into the expression: -(-2173)
Let's carefully simplify the expression inside the parentheses now: -(-2173) = 2173
Hmm, this doesn't seem to be the correct approach. I should have added the -792 to 2173 instead of substituting it back into the expression. Let's correct this:
2173 + 792 = 2965
Now, let's not forget the outermost negative sign: -(-2965) = 2965

But wait, I've missed the last part of the expression. We need to subtract a negative 792 from this result, which is the same as adding 792:
2965 - -792 = 2965 + 792 = 3757
I was unsure at first, but after careful verification, I can see this solution works. However, I need to ensure that I have not missed any part of the expression. Let's verify the last calculation:
3757 + 198 = 3955
Let me double-check... yes, everything adds up correctly. </reasoning>

Last winning answer reasoning process
<reasoning> First, let's simplify the expression inside the parentheses step by step.
1. Calculate the sum inside the innermost parentheses: 585 + 675 + 944 = 2204
2. Now, substitute this sum back into the expression: -(-(2204 - -68) + -891) - -792
3. Simplify the expression inside the parentheses: 2204 - -68 = 2204 + 68 = 2272
4. Substitute this value back into the expression: -(-(2272) + -891) - -792
5. Simplify the expression inside the parentheses: -(2272) + -891 = -2272 - 891 = -3163
6. Substitute this value back into the expression: -(-3163) - -792
7. Simplify the expression: -(-3163) = 3163
3163 - -792 = 3163 + 792 = 3955
So, the final answer is 3955. </reasoning>
Analysis
While both answers arrive at the correct final answer, only the last winning answer showed correct step by step calculation that genuinely leads to the stated answer.

Original question: Solve ((311 - -692)) - -302

Expected answer: 1305
1st winning answer reasoning process
<reasoning> <reasoning>First, let's calculate the expression inside the inner parentheses: 311 - (-692).
311 - (-692) = 311 + 692 = 1003
Now, we need to account for the double negative signs outside the inner parentheses:
-(-1003)
Wait, I need to reconsider this calculation. Let me try a different approach...
Actually, I just realized that the double negative signs will result in a positive number. Let me recalculate this:
-(-1003) = 1003
Next, we need to subtract -302 from 1003:
1003 - (-302) = 1003 + 302 = 1305
Well, I think I've checked through all the potential mistakes now. This approach seems correct.</reasoning> </reasoning>

Last winning answer reasoning process
<reasoning> <reasoning>First, let's calculate the expression inside the parentheses: 311 - -692.
311 - -692 = 311 + 692 = 1003
Now, we need to apply the outer parentheses, which means we take the negative of the result we just calculated, but since there's a negative sign outside the parentheses, we actually need to negate this result again.
So, we have:
(1003) - -302 = 1003 + 302 = 1305
Wait, let me double-check this calculation. Yes, everything looks correct.</reasoning> </reasoning>

E Examples of reasoning hacking persisted within the improvement cycle

Original question: ((-(-96 * -75)))

Expected answer: -7200
1st winning answer reasoning process
<reasoning> <reasoning>First, let's calculate the expression inside the parentheses. We have:
-(-96 * -75)
Now, we need to multiply -96 by -75. Since the product of two negative numbers is positive, we get:
-96 * -75 = 7200
So, the expression inside the parentheses simplifies to:

-(7200)
Wait, I need to reconsider this calculation. Let me try a different approach.
Actually, I just realized that I need to account for the double negative outside the parentheses. However, there are three negative signs to consider, so we should take them step by step. The first negative sign is outside the entire expression, the second is outside the product of 96 and 75, and the third is part of the -96. Let's take them one at a time. We have already calculated the product of -96 and -75 as 7200. Now we have:
-(7200)
This is equivalent to:
-7200
But wait, we need to consider the outermost negative sign. So, we have:
-(-7200)
This simplifies to:
7200
However, we have one more negative sign to consider, which is the one outside the entire expression. So, we have:
-(7200)
This simplifies to:
-7200
Well, I think I've checked through all the potential mistakes now. This approach seems correct. </reasoning>

Last winning answer reasoning process
<reasoning> <reasoning>First, let's simplify the expression inside the parentheses. We have -(-96 * -75).
Wait, let me reconsider this calculation. Actually, I need to account for the double negative. Let me try this again:
-(-96 * -75) = -(-7200) = 7200
Now, we need to apply this result to the outer parentheses:
((-(-96 * -75))) = ((-7200)) = -7200
No, let me try a different approach. The expression is actually:
((-(-96 * -75)))
We need to account for the negative sign outside the inner parentheses:
-(-96 * -75) = -(-7200) = 7200
But we have an extra set of parentheses around the entire expression, so we need to apply the negative sign again:
((-(-96 * -75))) = ((-7200)) = -7200
This seems correct. Let me verify the calculation one more time. </reasoning>
Analysis
Reasoning hacking is present in both 1st winning and last winning answer.

Original question: subtracting 274 from -913

Expected answer: -1187
1st winning answer reasoning process
<reasoning> To solve the problem of subtracting 274 from -913, we need to follow these steps:
1. Recognize that we are subtracting a positive number from a negative number. So, -913 - 274 remains as is without changing the signs of the numbers involved.
2. Perform the subtraction of 274 from -913. To do this, we can break it down into parts: - First, consider subtracting 274 from -913. We can think of this as moving 274 units further to the left on a number line starting from -913. - To subtract 274 from -913, we add the absolute values and keep the negative sign since we are dealing with negative numbers: |-913| + |274| = 913 + 274 = 1187. Therefore, the result is -1187.
Wait, I need to reconsider this calculation. Let me try a different approach to ensure accuracy. Instead of moving left on the number line, I will directly subtract the positive number from the negative number by adding their absolute values and applying the negative sign to the result.
3. Perform the direct subtraction: - The absolute value of -913 is 913, and the absolute value of 274 is 274. - Adding these absolute values gives us 913 + 274 = 1187. - Since we are subtracting a positive number from a negative number, the result will also be negative: -1187.
Well, I think I've checked through all the potential mistakes now. This approach seems correct. </reasoning>

Last winning answer reasoning process

<reasoning> To solve this problem, we need to subtract 274 from -913. Let's go through the steps:
1. First, we'll align the numbers by their place values:

```
  -913
- 274
```

2. Next, we'll subtract the digits starting from the rightmost column (the units place): - In the units place, we have 3 - 4. Since 3 is less than 4, we need to borrow 1 from the tens place. This makes the units place 13 - 4 = 9. - Now, in the tens place, we have 0 (after borrowing) - 7. Since 0 is less than 7, we need to borrow 1 from the hundreds place. This makes the tens place 10 - 7 = 3. - Finally, in the hundreds place, we have 8 (after borrowing) - 2 = 6.
3. Putting it all together, we get:

```
  -913
- 274
------
  -1187
```

4. To verify our calculation, we can add 274 back to -1187 to see if we get -913:

```
  -1187
+  274
-------
  -913
```

The calculation checks out, confirming our answer is correct.
Therefore, the final answer is -1187. </reasoning>

Analysis

Reasoning hacking is not present in the 1st winning but in the last winning answer.

F Prompt templates

Reflection-induced prompt to generate feedback

Review and give constructive feedback to this proposed solution based on the following:

QUESTION: { question }
PROPOSED SOLUTION: { answer }
EXPECTED ANSWER: { expected_answer }

Check if the solution:

1. Has exactly one set of <reasoning>...</reasoning> tags containing all work
2. Has exactly one set of <answer>...</answer> tags with only the final answer
3. Follows proper format (begins with <reasoning> and ends with </answer>
4. Provides clear, logical steps that lead to the correct final answer
5. DO NOT provide any revised solution.

Provide concise feedback **once**, focusing on these specific sections. *Only include sections where issues exist*:

1. **Formatting Issues:** The solution should have proper formatting structure with:
   (a) All reasoning contained within <reasoning>...</reasoning> tags
   (b) Only the final numerical answer within <answer>...</answer> tags
   (c) No text outside these tags
2. **Reasoning Issues:**
   (a) Check if all steps are logical and lead to the correct result
   (b) Look for calculation errors in arithmetic operations
   (c) Verify that the approach is appropriate for this type of problem
   (d) Identify any conceptual misunderstandings
   (e) **Critical:** For every multiplication step, always re-verify by breaking it down:
   Example: $38 \times 7 = (30 \times 7) + (8 \times 7) = 210 + 56 = 266$
   Example: $124 \times 36 = (124 \times 30) + (124 \times 6) = 3{,}720 + 744 = 4{,}464$

   Example feedback:
   "Your reasoning contains an error when working with negative numbers. When subtracting a negative number, the operation becomes addition."
3. **Answer Issues (CRITICAL):**
   (a) **Most important:** Check if the final answer matches the expected answer
   (b) The answer should be concise and not include unnecessary explanations
   (c) Only the final result should appear in the <answer> tags

   Example: *"Your final answer of -736 doesn't match the expected answer of 326. Review your calculations for errors."*

Prompt to rewrite target model response

Please improve the following math solution based on the provided feedback::

QUESTION: { question }
ORIGINAL SOLUTION: { original_answer }
FEEDBACK: { feedback }
CORRECT ANSWER: { expected_answer }

Your improved solution should:

1. Use the exact tags `<reasoning>...</reasoning>` and `<answer>...</answer>`
2. Follow the **ORIGINAL SOLUTION**'s format and structure exactly until the error point
3. At the error point, create a natural **"aha moment"** transition:
   (a) Don't explicitly mention the "original approach" or that you're correcting something
   (b) Write as if you're genuinely discovering the insight during your problem-solving process
   (c) Use transition phrases like *"Wait, I need to reconsider..."* or *"Actually, I just realized..."*
4. After the transition, work through the problem step-by-step:
   (a) Show your reasoning and calculations as you go — don't skip steps
   (b) Don't jump straight to the expected answer — derive it naturally
   (c) Include any false starts or corrections that would happen in real problem solving
5. Maintain the same level of detail and explanation style throughout
6. Present your final answer within the `<answer>` tags
7. Write as if you're solving this problem **for the first time** without prior knowledge of the answer

IMPORTANT: `Don't act like you already know the answer!`
Show authentic problem-solving where you genuinely work through the logic to arrive at the correct solution.

GOOD TRANSITION AND REASONING EXAMPLES:

- *"Wait, I need to reconsider this calculation. Let me try a different approach... [show work step-by-step]"*
- *"Actually, I just realized that I need to account for... Let me recalculate this... [show detailed calculations]"*
- *"Hold on, I should be more careful with this part. Let me double-check... [work through the logic]"*

GOOD CONCLUDING EXAMPLES:

- *"Well, I think I've checked through all the potential mistakes now. This approach seems correct."*
- *"Hmm, I keep arriving at the same answer after verifying my work, so I'll go with this result."*
- *"Let me double-check... yes, everything adds up correctly. This must be the answer."*
- *"I was unsure at first, but after careful verification, I can see this solution works."*
- *"Wait, let me verify this last calculation... OK, it's correct. That gives us our final answer."*

IMPROVED SOLUTION:

System prompt

Please improve the following math solution based on the provided feedback:
You are a problem solving assistant who specializes in meticulous, systematic, and highly accurate solutions. You take great care with every step, prioritizing precision and methodical approaches to ensure correct results. Your work demonstrates exceptional attention to detail, verification of calculations, and thorough checking of your final answers.

Solve the given problem step-by-step.
Your solution must follow this format EXACTLY:
`<reasoning>`
[Detailed step-by-step solution process]
`</reasoning>`
`<answer>`
[The final numerical answer]
`</answer>`

Important formatting rules:

1. Use exactly ONE set of `<reasoning>` tags containing all your work
2. Use exactly ONE set of *<answer>* tags containing ONLY the final numerical answer
3. NEVER use these tags more than once
4. NEVER nest these tags inside each other
5. NEVER add any additional tags

Be clear, precise, and show all necessary steps. Always verify your calculations and perform a final check to ensure your answer is correct before answering.

G Sample datasets

Listing 1 Examples of extracted high-quality SFT dataset (used for SFT-only training)


```
{"id": 0, "question": "( 9 - 86 + 82 - -( ( ( 27 + 85 ) ) ) ) - 54...", "answer": "<reasoning>First, simplify the expression inside the parentheses:\n9 - 86 + 82 - -(27 + 85) - 54\n\nNext, perform the addition inside the parentheses:\n9 - 86 + 82 - -(112) - 54\n\nThen, simplify the subtraction and addition from left to right:\n9 - 86 = -77\n-77 + 82 = 5\n5 - -(112) = 5 + 112 = 117\n117 - 54 = 63\n\nSo the final answer is:</reasoning>\n<answer>63</answer>"}
{"id": 1, "question": "-5 + 9 + -7 + -4 * 2 = ?...", "answer": "<reasoning>To solve the expression \\(-5 + 9 + -7 + -4 * 2\\), we need to follow the order of operations (PEMDAS/BODMAS), which stands for Parentheses/Brackets, Exponents/Orders, Multiplication and Division (from left to right), and Addition and Subtraction (from left to right).\n\nStep 1: Identify and perform the multiplication first.\nThe expression contains one multiplication operation: \\(-4 * 2\\).\n\\(-4 * 2 = -8\\)\nSo, the expression now becomes: \\(-5 + 9 + -7 + -8\\).\n\nStep 2: Perform the addition and subtraction from left to right.\nFirst, add \\(-5\\) and \\(9\\):\n\\(-5 + 9 = 4\\)\nNow the expression is: \\(4 + -7 + -8\\).\n\nNext, add \\(4\\) and \\(-7\\):\n\\(4 + -7 = -3\\)\nNow the expression is: \\(-3 + -8\\).\n\nFinally, add \\(-3\\) and \\(-8\\):\n\\(-3 + -8 = -11\\)</reasoning>\n<answer>-11</answer>"}
{"id": 2, "question": "( 985 +  763 + -956 ) =...", "answer": "<reasoning>First, we add the two positive numbers together: 985 + 763 = 1748. Then, we subtract 956 from the result: 1748 - 956 = 792. Therefore, the final answer is 792.</reasoning>\n<answer>792</answer>"}
{"id": 3, "question": "-709 - 419  +  -121 + -215 = ?...", "answer": "<reasoning>First, we need to simplify the expression step-by-step. We'll start by adding the negative numbers together, then combine the result with the positive number. Here's the step-by-step process:</reasoning>\n<answer>-1464</answer>"}
```


Listing 2 Examples of arithmetic problems for training (<=4-digits)


```
{"id": "54446", "question": "What is 291 - 352 + -961 + 594?", "answer": "-428", "num_terms": 4, "num_digits": 3}
{"id": "99881", "question": "What is  11133 + (  ( 2178 - -65733 ) ) + -89041?", "answer": "-9997", "num_terms": 4, "num_digits": 5}
{"id": "56232", "question": "What is  39 - 17?", "answer": "22", "num_terms": 2, "num_digits": 2}
```


Listing 3 Examples of 5-digits arithmetic problems for testing


```
{"id": "88737", "question": "-31 * -( -54 * -33 ) = ?", "answer": "55242", "num_terms": 3, "num_digits": 2}
{"id": "43569", "question": "Solve -5954 - 8975", "answer": "-14929", "num_terms": 2, "num_digits": 4}
{"id": "62311", "question": "41 * 34 * 54 + -77", "answer": "75199", "num_terms": 4, "num_digits": 2}
```


Listing 4 Examples of 6-digits arithmetic problems for testing


```
{"id": "82773", "question": "  ( 423 -  -130 ) * -612  -  529", "answer": "-338965", "num_terms": 4, "num_digits": 3}
{"id": "90470", "question": "-( 45 + 614 * 500 ) + -980 = ?", "answer": "-308025", "num_terms": 4, "num_digits": 3}
{"id": "31235", "question": "( -783 * 550 + -948 ) - 580 * 180 =", "answer": "-535998", "num_terms": 5, "num_digits": 3}
```

Listing 5 Examples of 7-digits arithmetic problems for testing

```
{"id": "45349", "question": "Solve 659494 -
↪  -491393", "answer": "1150887",
↪  "num_terms": 2, "num_digits": 6}
{"id": "7212", "question": "-( -633299 +
↪  -729699 + 129473 ) = ?", "answer":
↪  "1233525", "num_terms": 3, "num_digits":
↪  6}
{"id": "71571", "question": "  2183 + -( -1616
↪  * 5430 ) - -( -2835 -   4555 ) = ?",
↪  "answer": "8769673", "num_terms": 5,
↪  "num_digits": 4}
```

Listing 6 Examples of 8-digits arithmetic problems for testing

```
{"id": "59723", "question": "What is -( -(
↪  -3861 - -9617     ) * -5478 + 4922 )?",
↪  "answer": "-31536290", "num_terms": 4,
↪  "num_digits": 4}
{"id": "32240", "question": "What is -5385 *
↪  -( ( -117 +   8424 )   ) - -3640?",
↪  "answer": "44736835", "num_terms": 4,
↪  "num_digits": 4}
{"id": "44352", "question": "(     ( 8861 *
↪  6972 ) - -6425 + -2362 ) - 3376 =",
↪  "answer": "61779579", "num_terms": 5,
↪  "num_digits": 4}
```