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

Mengdie Flora Wang, Haochen Xie, Mun Young Kim, Baishali Chaudhury,
Meghana Ashok, Suren Gunturu, Sungmin Hong, Jae Oh Woo
AWS Generative AI Innovation Center
`{florawan, haochenx, munkim, baishch, meghasho, gunsuren, hsungmin, jaeohwoo}@amazon.com`

Abstract

Contemporary advancements in language model reasoning typically require computationally intensive reinforcement learning (RL) and massive datasets, creating 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" cycle (R^3). 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% → 88.6%), AIME'25@10 (20.0% → 30.0%) and LastLetterConcat (40.7% → 53.3%) problems. The model-agnostic nature of our R^3 framework is further demonstrated through substantial improvements when applied to Mistral and LLaMA-based models. Remarkably, these gains are achieved using mere 700 basic arithmetic training samples, in 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 with minimal resource overhead. Our R^3 pipeline also generates high-quality SFT data with high-fidelity reasoning traces as byproduct, further enabling scalable and annotation-free fine-tuning. Code is available.[1]

1 Introduction

Enhancing the reasoning capabilities of large language models (LLMs) remains a central challenge in natural language processing. Recent advances in reasoning capabilities have largely relied on reinforcement learning (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 policy-gradient RL with an interactive refinement loop that combines Direct Preference Optimization (DPO) and Supervised Fine-Tuning (SFT), eliminating the need for explicit reward modeling or policy rollouts. A model iteratively generates answers, receives corrections from a stronger teacher model, and learns to internalize these refinements.

Unlike existing iterative refinement approaches that rely on student models to critique and correct their own outputs (Madaan et al., 2023; Wang et al., 2024)—which may suffer from superficial self-reflection (Liu et al., 2025)—we employ a teacher model to ensure high-fidelity corrections within a structured Reflect-Rewrite loop. Importantly, this design choice is not a permanent constraint but a pragmatic response to current limitations in LLM self-evaluation (Woo et al., 2025). Our framework is explicitly forward-compatible: the modular architecture allows the teacher dependency to be progressively removed as student models mature, enabling seamless transition from guided distillation to fully autonomous self-improvement while preserving the core DPO+SFT training dynamics.

---

[1] `https://github.com/aws-samples/sample-for-reflect-rewrite-repeat`

Findings of the Association for Computational Linguistics: EACL 2026, pages 1341–1363
March 24-29, 2026 ©2026 Association for Computational Linguistics

Contributions. This work challenges the assumption that reasoning improvements require massive datasets or complex reinforcement learning. We introduce R^3, demonstrating that structured iterative refinement over foundational tasks achieves competitive performance with 100-fold data efficiency through deterministic teacher-guided corrections and cross-domain transfer from arithmetic-only supervision.

We make four principal contributions. First, 700 arithmetic examples suffice to improve mathematical, symbolic, and commonsense reasoning, challenging the prevailing assumption that complex reasoning requires complex training data. Second, we address superficial self-reflection (Liu et al., 2025) by linking introspective language to explicit error corrections, with manual analysis confirming genuine logical improvements rather than stylistic mimicry. Third, systematic ablations reveal DPO and SFT's synergistic roles in maintaining performance under increasing complexity, demonstrating that neither component alone suffices for robust generalization. Fourth, we demonstrate model-agnostic applicability across Qwen, Mistral, and LLaMA with extreme computational efficiency—requiring only a single GPU and hours of training—making advanced reasoning enhancement accessible to resource-constrained teams.

2 Related Work

2.1 From RL-Based Preference Optimization to Direct Preference Optimization

Reinforcement-learning 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 trajectories. Recent audits reveal these gains can be brittle and reward-model dependent(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), and Full-Step-DPO(Xu et al., 2025b)—replace policy-gradient updates with supervised comparison losses. Orthogonally, Lee et al.(Lee et al., 2024; Woo, 2023) demonstrate that uncertainty-aware reward modeling with Bayesian approximation substantially improves preference learning by enabling principled data curation, highlighting the importance of preference data quality. Our work combines DPO with lightweight SFT, eliminating the need for explicit reward modeling or policy rollouts.

2.2 Data Efficient Mathematical Reasoning

Most math-centric LLMs like JiuZhang 3.0(Zhou et al., 2024) and RedStar(Xu et al., 2025a) rely on millions of synthetic examples. Later studies pursue efficiency by mining high-quality instances (LIMO(Ye et al., 2025), s1(Muennighoff et al., 2025)) or distilling compact datasets(Lightman et al., 2023). R^3 achieves meaningful improvements with just 700 elementary arithmetic tasks and more importantly, it advances beyond data efficiency by showing that a principled, iterative refinement process can endow models with reasoning abilities that generalize across domains. This unsupervised cross-domain transfer represents a major step beyond prior data efficient methods focused on in-domain or stylistic alignment(Zhou et al., 2023) (more discussion in Appendix F).

2.3 Self-Verification and Error Correction

Recent work explores how models can detect and repair reasoning failures. Early approaches trained separate verifiers(Cobbe et al., 2021), while later methods integrated verification into generation: ReAct(Yao et al., 2023) interleaves CoT with tool calls, and 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) show LLMs can iteratively improve their answers without manual labels. These works typically rely on the student model to critique and correct its own output. However, recent research indicates that small models often suffer from Superficial Self-Reflection (SSR) (Liu et al., 2025), where introspective language co-occurs with flawed reasoning. Recent theoretical analysis also highlights fundamental limitations in self-evaluation: Woo et al.(Woo et al., 2025) propose a probabilistic VC framework showing that models exhibiting strong discriminative self-assessment tend to demonstrate diminished calibration fidelity, suggesting an inherent trade-off that motivates external guidance in refinement pipelines.

2.4 Student–Teacher Distillation Paradigms

Student–teacher transfer compresses reasoning skills effectively. Early works distilled hidden states (DistilBERT(Sanh et al., 2019), MiniLM(Wang et al., 2020)); later research

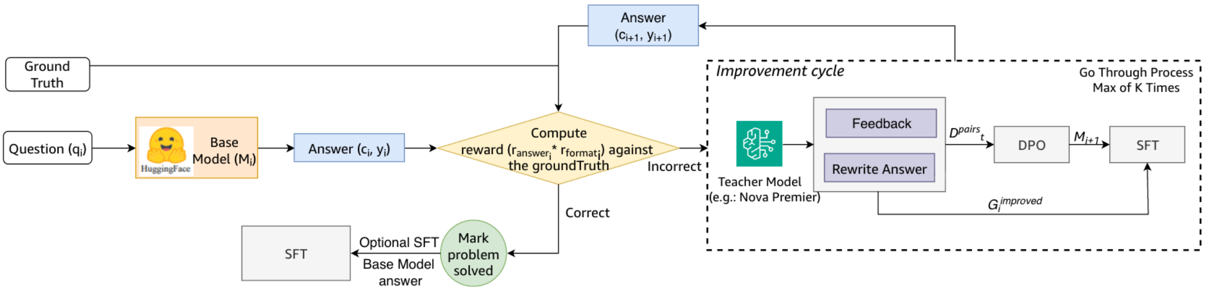


Figure 1: Our R^3 framework is inspired by effective one-to-one human tutoring pedagogy. Targeted feedback and rewritten answers from the teacher model guide the target model toward improved reasoning and problem solving through DPO and SFT training.

distilled CoT traces(Magister et al., 2022) or preference signals(Rafailov et al., 2023). TS-Align(Zhang et al., 2024) and AlpacaFarm(Madaan et al., 2023) show iterative teacher feedback can refine students with minimal manual labeling. We extend this by having a teacher model both *grade* and *rewrite* student answers, creating an automated tutor at negligible human cost.

These threads motivate our approach: unifying direct preference optimization, sample-efficient reasoning, superficial self-reflection, and student–teacher feedback into a streamlined framework that achieves meaningful reasoning improvements in small models through iterative refinement.

3 Method Overview

In this section, we outline our approach to optimizing the reasoning capabilities of small LLMs under computational constraints. As illustrated in Figure 1, our methodology comprises three primary components: (i) initializing the target model to generate original answers accompanied by chain-of-thought reasoning, (ii) leveraging an advanced teacher model and a DPO-based iterative improvement cycle to refine the target model's reasoning process, and (iii) verifying the target model's acquired competency over training samples and adopting 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 $M_0, M_1, \ldots, M_T$.

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

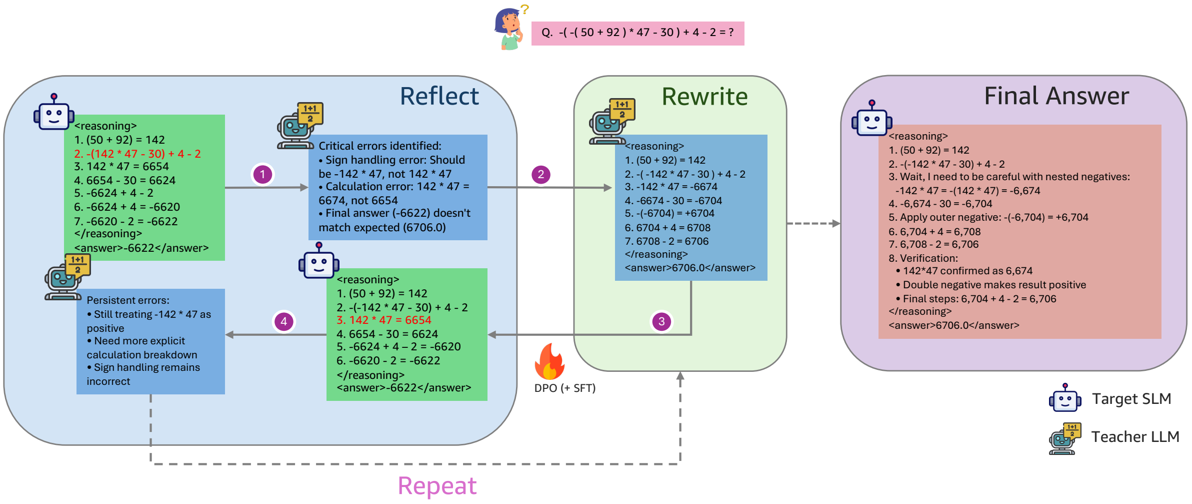
$$(c_i, y_i) \sim M_t(x_i) \text{ for all } x_i \in \mathcal{D} \qquad (1)$$

We compute the reward r_i for the initial response based on both answer correctness ($r_{\text{answer}_i} = \mathbb{I}(y_i = y_i^*)$) and format correctness ($r_{\text{format}_i} = \mathbb{I}(\text{valid tags})$), where valid tags require proper `<reasoning>` and `<answer>` structure. The total reward is $r_i = r_{\text{answer}_i} \cdot r_{\text{format}_i}$. If $r_i = 1$, we consider the example already mastered and require no further training. Otherwise, we enter an improvement cycle. This multiplicative design eliminates partial-credit noise, restricting DPO training to samples that are both mathematically correct and structurally valid.

3.2 Preference Optimization Improvement Cycle

Reflect and rewrite base response. This improvement cycle employs a teacher 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. The teacher 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 $\mathcal{D}_i^{\text{pairs}}$ using the base generation G_i^{base} from the current model M_t and improved generations G_i^{improved} from the teacher model. Given the preference pairs, we can now train a new model M_t that will become our next model M_{t+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, we ensure that we do not train on



Q. -(-(50 + 92) * 47 - 30) + 4 - 2 = ?

```
<reasoning>
1. (50 + 92) = 142
2. -(142 * 47 - 30) + 4 - 2
3. 142 * 47 = 6654
4. 6654 - 30 = 6624
5. -6624 + 4 - 2
6. -6624 + 4 = -6620
7. -6620 - 2 = -6622
</reasoning>
<answer>-6622</answer>
```

Critical errors identified:
- Sign handling error: Should be -142 * 47, not 142 * 47
- Calculation error: 142 * 47 = 6674, not 6654
- Final answer (-6622) doesn't match expected (6706.0)

```
<reasoning>
1. (50 + 92) = 142
2. -( -142 * 47 - 30 ) + 4 - 2
3. -142 * 47 = -6674
4. -6674 - 30 = -6704
5. -(-6704) = +6704
6. 6704 + 4 = 6708
7. 6708 - 2 = 6706
</reasoning>
<answer>6706.0</answer>
```

```
<reasoning>
1. (50 + 92) = 142
2. -(-142 * 47 - 30) + 4 - 2
3. 142 * 47 = 6654
4. 6654 - 30 = 6624
5. -6624 + 4 – 2 = -6620
6. -6620 - 2 = -6622
</reasoning>
<answer>-6622</answer>
```

Persistent errors:
- Still treating -142 * 47 as positive
- Need more explicit calculation breakdown
- Sign handling remains incorrect

```
<reasoning>
1. (50 + 92) = 142
2. -(-142 * 47 - 30) + 4 - 2
3. Wait, I need to be careful with nested negatives:
  -142 * 47 = -(142 * 47) = -6,674
4. -6,674 - 30 = -6,704
5. Apply outer negative: -(-6,704) = +6,704
6. 6,704 + 4 = 6,708
7. 6,708 - 2 = 6,706
8. Verification:
  • 142*47 confirmed as 6,674
  • Double negative makes result positive
  • Final steps: 6,704 + 4 - 2 = 6,706
</reasoning>
<answer>6706.0</answer>
```

Figure 2: An example of the R^3 process showing how the target model progressively refines its reasoning under the guidance of the teacher model. The figure traces the evolution from an initial incorrect solution with sign handling errors (left) through guided rewriting (middle) to a correct solution with proper verification (right). This concrete example demonstrates how our framework identifies and corrects mathematical errors through structured feedback and multiple refinement iterations.

examples with incorrect formatting. Given the critical role of the teacher model, it must not only solve tasks with near-perfect accuracy but also produce well-structured Chain-of-Thought (CoT) reasoning. This ensures that the *Rewrite* step provides valid ground truth for the student to internalize. Figure 2 provides a concrete example of this process, showing how a target model progressively improves its reasoning on a mathematical problem through multiple reflection and rewriting iterations.

3.3 Learning Verification & Reinforcement

Learning verification. We evaluate the trained model M_T 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 $\left(r_0^{M_T} = 1\right)$. 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. As shown in Figure 2, this iterative refinement can transform initially flawed reasoning with sign handling errors into a correct solution with proper verification steps.

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.

The DPO and SFT process maximizes the following objectives respectively:

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

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

where q represents input questions, o^+ and o^- are preferred and non-preferred responses respectively, π_θ is the current model policy being optimized, π_{ref} is the previous iteration's policy in R^3, and β controls the preference signal strength.

4 Experiments

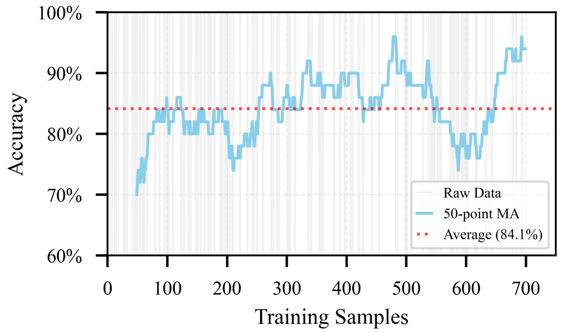
4.1 Experimental Setup

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 explained in the section below. 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 teacher models. Training was conducted on 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. We adopt LoRA not merely for computational efficiency, but as an implicit regularizer to mitigate overfitting in our low-data regime (700 samples). Detailed experimental settings are reported in Appendices B.

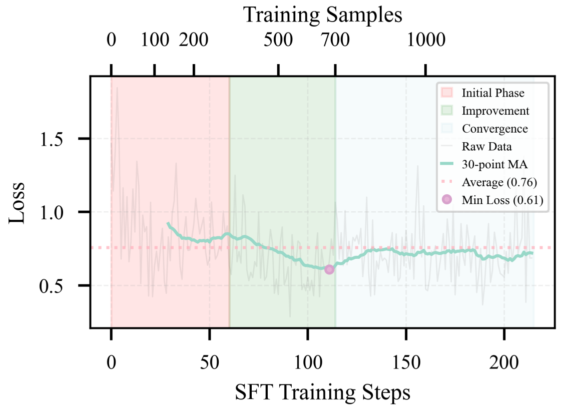
4.2 Training Sample Construction

Our training samples are derived from the publicly available Tiny-GRPO dataset [2], which contains 100,000 arithmetic problems. To construct our training set, we filtered this dataset to include only problems with 1- to 4-digit answers, yielding approximately 45,000 candidates. A subset of 4-digit problems was reserved as a held-out test set; the remainder formed the training pool. Samples were drawn randomly, and we monitored first-attempt accuracy on a disjoint evaluation set to assess generalization (Figure 3a). Training was terminated when performance gains plateaued at 700 examples (Figure 3b).

The complete R^3 pipeline—including reflection, teacher correction, and optimization—was applied exclusively to these samples. Notably, curation effort focused on generating high-quality teacher reflections rather than hand-selecting specific problems, enabling us to study how structured feedback impacts learning under strict data constraints. The 700-sample threshold reflects an empirical stopping point based on performance saturation, consistent with our emphasis on data efficiency under minimal supervision. Example training samples are provided in Appendix M (Listing 2).



(a) First time accuracy during R^3 improvement cycle: This metric measures whether the model correctly solves a problem on its first attempt when encountering a new question. It serves as a meaningful proxy for test accuracy, demonstrating significant improvement over training cycles.



(b) SFT loss during R^3 improvement cycle showing three distinct phases: Initial Phase (high variance), Improvement (steady decline), and Convergence (stable performance). The loss reaches minimum around 700 samples, demonstrating that our training set size is sufficient for effective learning without requiring additional data.

Figure 3: Training metrics during the R^3 improvement cycle showing the progression of model performance and learning efficiency.

4.3 Benchmark Datasets and Metrics

To evaluate the reasoning capabilities of our small LLMs, we selected six mathematics-focused popular benchmarks: GSM8K (Cobbe et al., 2021), SVAMP (Patel et al., 2021), AIME'24 (HuggingFaceH4, 2024), AIME'25 (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 used Pass@1 metrics for larger datasets and Pass@10 for smaller benchmarks (AIME'24, AIME'25, AMC'23). Pass@1 measures the proportion of problems correctly solved on first attempt without examples, emphasizing small LLMs' inde-

---

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

pendent reasoning ability. However, this metric introduces high volatility for small benchmarks—in AIME'24 and AMC'23 (30-40 examples each), a single question affects Pass@1 by over 3 percentage points. Following best practices (Hochlehnert et al., 2025), we report Pass@10 for smaller datasets to reduce variance and ensure robust evaluation. Our evaluations used the Evalchemy framework with vLLM backend (Guha et al., 2024) across ten random seeds. We configured model-appropriate parameters, including system prompt chat templates, temperature of 0.7, and adjusted maximum token lengths (4096 for QwenMath-based models; 8192 for other models).

4.4 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) their instruction fine-tuned versions: Qwen 2.5 7B Instruct and Qwen 2.5 Math 7B Instruct; (iii) comparable state-of-the-art models from prior work: 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 R^3 pipeline. To validate that our framework's effectiveness stems from the methodology rather than a specific teacher model, we conducted experiments with two distinct high-performing models: Amazon Nova Premier and DeepSeek-R1. Both demonstrate near-perfect task accuracy, robust CoT generation, and reliable instruction following (detailed metrics and prompts in Appendices H and L).

Table 1 presents comprehensive results with best and second-best performances highlighted. Our R^3 pipeline improves the general-purpose Qwen 2.5 7B across six of eight benchmarks, with gains of 5.5, 2.6, 10, 2.4, 12.6 and 1 percentage points on GSM8K, SVAMP, AIME'25, MATH 500, LLC and CSQA respectively. These improvements demonstrate that even simple arithmetic training can enhance performance on diverse reasoning tasks. The instruction-tuned versions of the target models underwent extensive optimization with over 1 million SFT samples and sophisticated multi-stage reinforcement learning (offline DPO, online GRPO), representing a reasonable upper bound for their model families. Notably, our R^3-trained models achieve comparable or better performance on four of eight tasks (GSM8K, SVAMP, AIME'25, AMC'23) despite substantially less training data.

To demonstrate the model-agnostic nature of the R^3 pipeline, we evaluated JiuZhang3.0 7B (Mistral-based) and JiuZhang3.0 8B (LLaMA-based) before and after applying our pipeline. While these models show strong out-of-the-box performance on standard mathematical tasks such as GSM8K and SVAMP, they significantly underperform on more complex reasoning tasks (AIME'25 and AMC'23) and broader symbolic and cognitive tasks (LLC and CSQA). After applying our R^3 pipeline, we observe substantial improvements across all benchmarks: 1-21.9 percentage points on GSM8K, 3.3-6.7 percentage points on AIME'24, 6.7-13.3 percentage points on AIME'25, 17.5-27.5 percentage points on AMC'23, 9.2-14.6 percentage points on MATH 500, 16.0-20 percentage points on LLC, and 6.5-27.2 percentage points on CSQA. These consistent improvements across diverse model architectures and benchmarks demonstrate that our R^3's effectiveness in enhancing reasoning capabilities regardless of underlying model architecture.

4.5 Ablation Study

To gain deeper insight into three key components of the R^3 pipeline - structured reflection, the joint synergy of DPO+SFT, and multi-cycle refinement, we conduct ablation experiments across two settings: (i) generalization to arithmetic problems exceeding training complexity , and (ii) zero-shot performance on established 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.

4.5.1 Structured Reflection

Recent analyses have raised questions about superficial self-reflection in RL pipelines, where models generate introspective phrases like *rethink* without actual logical improvement (Liu et al., 2025). This phenomenon, termed *Superficial Self-Reflection* (SSR), describes cases where reflective language co-occurs with flawed reasoning, and response length does not reliably indicate cognitive depth.

To address these limitations, we introduce structured reflection with teacher guidance in our R^3 framework. A stronger teacher model (Amazon Nova Premier) provides targeted corrections for erroneous reasoning paths, which are then used for DPO and SFT. This process enforces consistent

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
Based 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 benchmark datasets and two cognitive reasoning datasets. For smaller datasets (AIME'24, AIME'25, AMC'23), we report Pass@10 metrics, while larger datasets use Pass@1. Bold numbers indicate the best performance within each model group. Note on **Qwen 2.5 Math 7B**: This model demonstrated inconsistent adherence to our evaluation protocol, specifically failing to consistently produce responses within the required `<reasoning>...</reasoning>` and `<answer>...</answer>` tag structure (format non-compliance), leading to artificially deflated accuracy across benchmarks. This formatting issue—rather than reasoning capability—accounts for its unexpectedly low baseline performance. In contrast, Qwen 2.5 7B reliably followed the protocol, enabling fair performance measurement.

reasoning improvement through transparent and verifiable training signals, free from reward-based ambiguity.

Structured out-of-distribution generalization: arithmetic by digit length. We assess generalization to problems exceeding training complexity. While training includes only up to 4-digit answers, we evaluate on 4-8 digit problems, representing controlled OOD generalization where numerical complexity increases beyond the training regime. As shown in Table 3, structured reflection consistently outperforms no-reflection variants: 8.5 percentage points improvement on 4-digit problems, 2.5 on 5-digit, and 0.5-1 on 6+ digit problems. These results confirm that simple arithmetic supervision with structured reflection enables generalization beyond the training distribution.

Out-of-distribution generalization: mathematical and cognitive benchmarks. We extend evaluation to three mathematical benchmarks (GSM8K, SVAMP, MATH500) plus two cognitive benchmarks (LLC and CSQA). Table 2 shows consistent gains with reflection: 18 percentage points on LLC and 6 on MATH500. Our current framework uses teacher guidance to generate improved response as a pragmatic choice given current small LLMs' limited self-evaluation capabilities. However, the modular nature of the framework remains forward-compatible for future autonomous self-improvement. For reproducibility, Appendix L includes reflection prompts and Appendices I and J shows learning progressions.

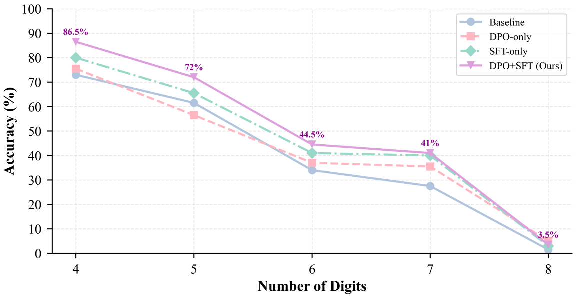


Figure 4: Qwen 2.5 7B accuracy across digits 4-8. DPO+SFT show most performance gains over the target model followed by SFT-only on the last winning answer.

4.5.2 DPO + SFT Synergy

In our framework, the integration of DPO and SFT emerges naturally from the reflection-rewrite process. The teacher-guided rewritten outputs generate structured preference pairs that function as high-fidelity training signals for DPO, while SFT consolidates these patterns into the model's policy. The novelty of our approach lies not in using DPO or SFT individually, but in their emergent synergy: the reflection loop transforms unstructured feedback into precise alignment signals with clear causal attribution. This differentiates our methodology from prior work such as LIMA (Zhou et al., 2023), InstructGPT (Ouyang et al., 2022), and RLHF-based approaches (Bai et al., 2022), which rely on externally curated preference data (further analysis in Appendix F).

DPO+SFT consistently outperforms DPO-only

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	**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 Attempt 1	88	87.7	57.8	52.7	**71.8**
Qwen 2.5 7B - Nova Premier R^3 - Max Attempt 5	**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 teacher 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	**86.5**	**72.0**	**44.5**	**41.0**	**3.5**

Table 3: Structured reflection ablation study across 4-8 math digits dataset - (Amazon Nova Premier as the teacher 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 teacher model is Amazon Nova Premier in both cases.

and SFT-only baselines across both arithmetic tasks exceeding training complexity and established reasoning benchmarks. As illustrated in Figure 4, while all approaches show declining gains as problem complexity increases, DPO+SFT maintains substantially higher performance across digit lengths when applied to Qwen 2.5 7B. This pattern holds across diverse benchmarks and model architectures, as detailed in our comprehensive ablation studies (See Appendix C). In summary, DPO and SFT play complementary roles: DPO drives preference-based reasoning refinement, while SFT stabilizes and generalizes the learned behaviors. Together, they form a synergistic training loop that yields stronger generalization and more coherent reasoning than either component alone.

4.5.3 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 or inconsistent logical processes. 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 with multi-cycle refinement consistently outperform those trained on their initial correct outputs across benchmarks such as GSM8K, SVAMP, MATH 500 and LLC.

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. Appendices I and J provides concrete examples of how our multi-cycle refinement transforms initially flawed reasoning into coherent logical processes.

4.6 Further Extension to Symbolic Logic Supervision

The arithmetic-based improvements observed thus far raise a fundamental question: do these gains stem from numerical pattern recognition, or do they reflect deeper structural reasoning capabilities? To test the numeric grounding hypothesis—whether arithmetic supervision is necessary for reasoning transfer—we extend our evaluation to a domain devoid of numerical computation: symbolic logic. We present this controlled experiment, where we replace arithmetic tasks with first-order logic problems to isolate the role of symbolic structure from numeric grounding.

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). NCD measures the similarity between two strings by comparing their compressed lengths, with lower values indicating greater similarity. During training, outputs were treated as correct when their NCD to the reference solution fell below a threshold of 0.4, determined through preliminary validation.

Table 5 compares the zero-shot performance of the symbolic logic-trained model with its arithmetic-trained counterpart across six mathematical benchmarks. While the symbolic logic model performs competitively, it consistently lags behind the arithmetic-trained variant across all benchmarks, with performance gaps ranging from 1.4 to 17.5 percentage points.

Training Source	**GSM8K**	**SVAMP**	**AIME'24@10**	**AIME'25@10**	**AMC'23@10**	**MATH500**
R^3 - Arithmetic (Ours)	**88.3**	**88.3**	**23.3**	**16.7**	**75.0**	**63.6**
R^3 - Symbolic Logic (Ours)	85.6	85.7	16.7	13.3	57.5	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@10 is used here for comparison 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, and MATH500 remain competitive (e.g., 2.7 percentage points lower for GSM8K and 1.4 for MATH500), 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 (-3.4) and AMC'23 (-17.5) suggests a limitation in symbolic logic's inductive reach, likely due to task-specific symbolic manipulations. These results emphasize the importance of structural alignment between supervision and target tasks: symbolic logic can scaffold general reasoning, but arithmetic grounding is essential for advanced mathematical performance. Future work may explore hybrid curricula combining symbolic, arithmetic, and commonsense reasoning to achieve broader and more robust cross-domain 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. The model-agnostic nature of R^3 is further demonstrated by consistent gains when applied to Mistral and LLaMA-based models, underscoring its broad applicability.

Beyond performance improvements, the R^3 framework produces high-quality, step-by-step reasoning traces (Appendix M, Listing 1) that serve as reusable supervision data, supporting both fine-tuning and evaluation in future work. The modular design 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..

Our findings suggest that the path to advanced reasoning may lie not in scale alone, but in the strategic application of structured supervision over foundational principles. Extending this approach to broader task categories may provide a path toward more accessible and scalable reasoning alignment for the wider community.

Limitations

Our approach has several limitations that suggest directions for future work. First, structured feedback currently relies on teacher models due to existing LLMs' limited self-evaluation capabilities; our modular design supports future transition to autonomous self-improvement as these capabilities mature. Second, while our method demonstrates consistent improvements across diverse architectures (Qwen, JiuZhang, LLaMA), further investigation into how base model characteristics affect improvement potential would provide valuable insights. Third, our evaluation focuses primarily on mathematical and symbolic reasoning; validation on broader domains such as commonsense inference and multi-step planning would more comprehensively establish the framework's generality. Despite these limitations, our core contribution remains robust: a data efficient approach that achieves significant improvements with orders of magnitude less data than traditional RL methods.

References

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, and 1 others. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

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*.

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.

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*.

JoonHo Lee, Jae Oh Woo, Juree Seok, Parisa Hassanzadeh, Wooseok Jang, JuYoun Son, Sima Didari, Baruch Gutow, Heng Hao, Hankyu Moon, Wenjun Hu, Yeong-Dae Kwon, Taehee Lee, and Seungjai Min. 2024. Improving instruction following in language models through proxy-based uncertainty estimation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 10955–10972. PMLR.

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*.

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.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

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.

Jae Oh Woo. 2023. Active learning in bayesian neural networks with balanced entropy learning principle. In *The Eleventh International Conference on Learning Representations*.

Jae Oh Woo, Mengdie Flora Wang, Rahul Ghosh, Baishali Chaudhury, and Mun Young Kim. 2025. Can LLMs reliably evaluate themselves? a probabilistic VC framework. In *NeurIPS 2025 Workshop on Bridging Language, Agent, and World Models for Reasoning and Planning*.

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. 2024. 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.

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.

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 R^3 Algorithm

Algorithm 1 Iterative Model Improvement with Preference Optimization and SFT

```
Require: Initial target model $M_0$; teacher model $R_{\text{Teacher}}$; training set $\mathcal{D} = \{(x_i, y_i^*)\}_{i=1}^N$
Initialize model $M \leftarrow M_0$, iteration counter $t \leftarrow 0$
for all questions $x_i \in \mathcal{D}$ do
    # Initial Evaluation Phase
    Generate response $(c_i, y_i) \sim M_t(x_i)$
    Compute answer reward $r_{\text{answer}_i} = \mathbb{I}(y_i = y_i^*)$
    Compute format reward $r_{\text{format}_i} = \mathbb{I}(\text{valid tags})$
    Total reward $r_i \leftarrow r_{\text{answer}_i} \cdot r_{\text{format}_i}$
    if $r_i \neq 1$ then
        Generate improved response $G_i^{\text{improved}} \sim R_{\text{Teacher}}(x_i)$
        Construct preference pair $\mathcal{D}_t^{\text{pairs}} = (G_i^{\text{base}}, G_i^{\text{improved}})$
        Initialize $\theta \leftarrow \text{Params}(M_t)$ ▷ Extract parameters from $M_t$
        Update $\theta$ using DPO loss on $\mathcal{D}_t^{\text{pairs}}$
        Update model $M_{t+1} \leftarrow M_\theta$
        (Optional) Perform SFT using $G_i^{\text{improved}}$, $M_{t+1} \leftarrow M_\theta$ with $\theta \leftarrow \text{Params}(M_{t+1})$
        $t \leftarrow t + 1$
    end if
    # Max $K$-step Verification Phase
    for iteration $k = 1, \ldots, K$ do
        Generate response $(c_i, y_i) \sim M_t(x_i)$
        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 $y_i$, $M_{t+1} \leftarrow M_\theta$ with $\theta \leftarrow \text{Params}(M_t)$
            $t \leftarrow t + 1$
            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
```

B Experiments Setting

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.

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 HuggingFace TRL framework unless otherwise specified.

C Ablation Study on DPO / SFT / DPO+SFT

Our proposed method combining DPO and SFT with multi-cycle R^3 approach outperformed baseline, DPO-only, and SFT-only configurations across various benchmarks. Table 7 summarizes results on 4-8 digit problems for both Qwen 2.5 7B and Qwen 2.5 Math 7B models. Since our training data only included problems with fewer than 4 digits, these results demonstrate generalization capabilities to more complex unseen problems. The DPO+SFT combination consistently showed superior performance on 4-7 digit problems compared to individual approaches, aligning with previous findings on RL+SFT benefits (Chu et al., 2025).

Notably, performance gains for 8-digit problems were minimal and inconsistent across approaches. We speculate this might be due to the original model's inherent limitations when handling highly complex calculations. Future research should explore the boundaries of generalization by including training data with higher digit counts (>4 digits), which would help determine the upper limits of model improvement across different fine-tuning approaches.

We further evaluated these variants on six mathematical reasoning benchmarks: GSM8K, SVAMP, AIME'24, AIME'25, AMC'23 and MATH 500. Similar to the main section, we report pass@10 on small-scale benchmarks in Table 7 for stable comparisons. DPO+SFT achieved the highest overall accuracy, consistently ranking the best on five of the six benchmarks. SFT-only showed surprisingly strong results when trained on refined responses, while DPO-only performance varied more across benchmarks, excelling on GSM8K and MATH 500 but underperforming on AIME and AMC datasets.

Takeaway. DPO and SFT play complementary roles—DPO emphasizes preference learning and logical ranking, while SFT ensures format and answer consistency. Together, they create a synergistic training loop that produces stronger generalization than either component alone.

D Teacher Distillation vs. Structured Refinement

To isolate the contribution of structured refinement from teacher knowledge transfer, we compare R^3 against direct teacher distillation via supervised fine-tuning.

Experimental Setup. We instructed the teacher model (Amazon Nova Premier) to generate solutions for the same 700 training samples using identical system prompts. We then fine-tuned Qwen 2.5 Math 7B using SFT-only on this teacher-generated data, creating a "Teacher-Distilled" baseline without the iterative DPO loop. This isolates the contribution of the structured Reflect-Rewrite process from teacher knowledge transfer alone.

Results. Table 8 presents the comparison. Both approaches dramatically improve over the base model, confirming the value of teacher guidance. However, the two methods exhibit distinct strength profiles.

On standard benchmarks (GSM8K, SVAMP), both methods achieve comparable performance, with differences within 1 percentage point. However, R^3 shows clear advantages on more challenging tasks: 10.0 percentage points higher on AIME'24, 6.7 percentage points higher on AIME'25, and 13.0 percentage points higher on MATH 500. The gains extend to symbolic and cognitive reasoning, with R^3 outperforming SFT-only distillation by 2.4 percentage points on LLC and 2.5 percentage points on CSQA.

Analysis. The divergent performance profiles suggest that R^3 and direct distillation develop different capabilities. SFT-only distillation excels at pattern matching for well-represented problem types (e.g., GSM8K-style word problems), while R^3 shows stronger generalization to complex, out-of-distribution tasks. We attribute this to the contrastive learning signal in DPO: by learning from preference pairs that contrast the student's errors against teacher corrections, R^3 develops more robust reasoning patterns that transfer beyond the training distribution.

E Performance Analysis by Model Strength

Our results show differential improvement magnitudes across baseline model capabilities. While absolute gains on stronger baselines are smaller than on weaker ones, we argue this pattern reflects fundamental properties of model optimization near performance ceilings rather than framework limitations.

Gains at the reasoning frontier. Improving models already approaching state-of-the-art performance in their parameter class presents inherently greater challenges than enhancing weaker

Model	Arithmetic 4 / 5 / 6 / 7 / 8 digits	GSM8K	SVAMP	AIME'24@10	AIME'25@10	AMC'23@10	MATH 500
Based on: Qwen 2.5 7B							
Qwen 2.5 7B DPO only	75.5 / 56.5 / 37 / 35.5 / **5**	87.5	86.7	10	3.3	42.5	**66.2**
Qwen 2.5 7B SFT only	80 / 65.5 / 41 / 40 / 3	87.6	87	6.7	6.7	45	65
Qwen 2.5 7B DPO + SFT	**86.5 / 72 / 44.5 / 41** / 3.5	**88.3**	**88.3**	**23.3**	**16.7**	**75**	63.6
Based on: Qwen 2.5 Math 7B							
Qwen 2.5 Math 7B DPO only	76 / 67.5 / 36.5 / 33.5 / 3	81.4	80.3	3.3	6.7	42.5	65.2
Qwen 2.5 Math 7B SFT only	81.5 / 75 / 50.5 / 43 / **5.5**	80.4	87.3	30	20	77.5	70.4
Qwen 2.5 Math 7B DPO + SFT	**91.5 / 85 / 55 / 44.5** / 3	**90.1**	**92**	**36.7**	**30**	**80**	**71.6**

Table 7: Accuracy across mathematical reasoning benchmarks - ablation study (Amazon Nova Premier as the teacher model). The best and second-best ones are marked in bold and underlined respectively.

Model	GSM8K	SVAMP	AIME'24@10	AIME'25@10	AMC'23@10	MATH 500	LLC	CSQA
Qwen 2.5 Math 7B (Base)	48.6	52.3	33.3	13.3	65.0	6.0	0.0	0.0
SFT-Only (Teacher Distillation)	90.5	91.3	26.7	23.3	**82.5**	58.6	30.3	52.1
R^3 (Ours)	90.1	**92.0**	**36.7**	**30.0**	80.0	**71.6**	**32.7**	**54.6**

Table 8: Comparison between SFT-only teacher distillation and R^3. Both use identical training data (700 samples) and the same teacher model (Amazon Nova Premier).

baselines. On AIME'25, a benchmark representing the upper bound of mathematical reasoning for models of this scale, our method improved Qwen 2.5 7B from 20.0% to 30.0%. This 10 percentage point absolute gain represents a 50% relative improvement—a substantial advance considering that surpassing 30% accuracy on AIME typically requires either significantly larger model scales or extensive reinforcement learning with human feedback. The 12.6 percentage point improvement on the symbolic LastLetterConcat task (40.7% → 53.3%) further demonstrates that gains extend beyond mathematical pattern matching to fundamental reasoning enhancement, confirming effectiveness even on stronger baselines.

Consistency with diminishing returns. The observed pattern—larger gains on weaker models and smaller gains on stronger ones—aligns with the principle of diminishing returns widely documented in machine learning optimization. Stronger base models such as Qwen 2.5 7B have already internalized many elementary reasoning patterns through pre-training and instruction tuning, leaving primarily complex edge cases and subtle reasoning failures to address. Consequently, each percentage point improvement requires progressively more refined interventions. Importantly, our method maintains double-digit percentage point gains even on highly capable models (e.g., +10.0%p on AIME'25), demonstrating genuine scalability to high-performance regimes rather than mere correction of elementary errors in weaker systems. This consistency across capability levels—from JiuZhang's substantial gains (+21.9%p GSM8K, +13.3%p AIME'25) to meaningful improvements on near-optimal Qwen models—confirms the robustness of our structured refinement approach across the model capability spectrum.

F Comparing R^3 to Other Data Efficient Mathematical Reasoning Approaches

Recent literature has explored various data efficient approaches for enhancing mathematical reasoning in language models (Zhou et al., 2023; Ouyang et al., 2022; Bai et al., 2022). In this section, we provide a comparative analysis between our R^3 framework and these existing methods, highlighting the fundamental differences in objectives, mechanisms, and outcomes.

While R^3 shares the goal of data efficiency with methods like LIMA (Zhou et al., 2023), its conceptual foundations and operational principles are distinctly different. LIMA focuses on alignment through stylistic imitation, demonstrating that a small number of curated examples can guide models toward user preferences. In contrast, R^3 is designed to induce reasoning generalization through a structured, failure-driven reflection loop. Our goal is not stylistic alignment but robust improvement in step-by-step problem solving, particularly in low-resource settings.

Aspect	LIMA (Zhou et al., 2023)	R^3 (Ours)
Primary goal	Stylistic alignment	Reasoning generalization
Training mechanism	Direct imitation	Reflection-rewrite loop
Cross-domain transfer	Limited	Substantial
Training examples	1,000 curated	700 arithmetic only
Domains improved	Within training domains	Arithmetic, symbolic, commonsense

Table 9: Comparison between LIMA and our R^3 approach

The mechanism underlying cross-domain transfer in R^3 appears to stem from the acquisition of general reasoning patterns rather than task-specific knowledge. Through structured reflection on foundational arithmetic tasks, the model learns to:

1. Decompose complex problems into manageable steps
2. Verify intermediate results
3. Detect and correct errors in reasoning chains

These meta-cognitive skills transfer beyond the training domain. While LIMA demonstrates that small data can align models superficially, R^3 shows that a small set of foundational tasks can trigger broader reasoning capabilities, transferring across arithmetic, symbolic manipulation (LastLetterConcat), and commonsense reasoning (CSQA)—domains far beyond the training scope.

In conclusion, the primary contribution of this work extends beyond mere data efficiency. Rather, we demonstrate that through a principled, iterative refinement process, models can develop generalizable reasoning capabilities that transfer across diverse domains. This cross-domain generalization advances beyond existing data efficient approaches confined to in-domain or stylistic alignment.

G Cost Analysis

Models	#API Calling (Token)			#Server Time (Hour)		Total Expenses
	Input	Output	Data Num.	Synthesizing	Training	
JiuZhang3.0 7B	300	877	10K	14	10	8,480 USD
Our method*	2132	701	1230	-	7	21.15 USD

Table 10: The baseline methods cost estimation details can be found here (Zhou et al., 2024)

To highlight the cost-efficiency of R^3, we provide an estimate of the end-to-end cost of training. Our method uses teacher model responses (Amazon Nova Premier or DeepSeek-R1 via Amazon Bedrock) to refine failed outputs. We measured the average token usage per iteration:

- Input tokens: 2000 (includes system prompt, input, feedback, and improvement prompt)
- Output tokens: 700

At each training session, the number of improvement steps was ranged from 500 to 700. Assuming 600 iterations, the total API invocation cost with Amazon Nova Premier is \$8.12 (on-demand pricing). For training, we used a g6e.xlarge EC2 instance, with total runtime of 6 hours, costing \$13.03. Thus, the total training cost is estimated at \$21.15, including both API and compute.

The substantially low training cost of our proposed method highlights the efficiency of our proposed method that how much the method can improve the performance on a given task with very small cost. However, we would like to note that the cost comparison for our proposed method is not apple-to-apple to the training cost of the other models shown in Table 10 since we trained our model with a small subset of training data on top of the pre-trained and/or fine-tuned model.

While our approach includes iterative loops, the overall cost remains orders of magnitude lower than full-scale training pipelines, due to the small number of examples and targeted refinement strategy. We acknowledge this is not a direct apples-to-apples comparison, as the baselines are often pre-trained from scratch or fine-tuned on large datasets, whereas our method operates on a small subset on top of an already-pretrained model. Overall, cost-efficiency is a key strength of R^3, demonstrating strong performance gains with minimal resource investment.

H Teacher Model Performance on Math Digit Dataset

Table 11 presents a comprehensive performance comparison among baseline models, our proposed method variants and teacher models on the digit generalization benchmark, allowing direct assessment of all evaluated models.

Model	**4-digit**	**5-digit**	**6-digit**	**7-digit**	**8-digit**
Qwen 2.5 7B	73.0	61.5	34.0	27.5	1.5
Qwen 2.5 7B DPO + SFT (Ours)	86.5	72.0	44.5	41.0	3.5
Nova Premier (Teacher)	**98.5**	**97.5**	88.0	80.5	47.0
Deepseek-R1 (Teacher)	**98.5**	96.5	**93.5**	**89.5**	**88.0**

Table 11: Performance comparison across different digit lengths

Both teacher models significantly outperform the base model, justifying their role in our teacher-guided refinement pipeline. Additionally, our R^3 method substantially improves the base model's performance—closing a meaningful portion of the gap to the teacher models with only 700 training examples. The gains (10–15% absolute on average except for 8-digit) demonstrate the effectiveness of R^3 in transferring reasoning capabilities from larger models into smaller ones—without reinforcement learning or large-scale data.

I 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.

J 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>

K 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.

L 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.

M 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\\)</reasonin⌋
↪  g>\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}
{"id": "26857", "question": "823   + 276 = ?",
↪  "answer": "1099", "num_terms": 2,
↪  "num_digits": 3}
{"id": "28204", "question": "Solve (   -( -11
↪  + 59 ) ) * 69", "answer": "-3312",
↪  "num_terms": 3, "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}
```