Hint-Assisted Reasoning: Improving Mathematical Problem Solving in Small Language Models

Anonymous ACL submission

Abstract

Small language models often struggle with complex mathematical reasoning due to limited capacity to maintain long chains of intermediate steps and to recover from early errors. We address this challenge by introducing a hint-assisted reasoning framework that incrementally guides small language models through multi-step mathematical problem solving. Our approach decomposes solutions into sequential reasoning steps and provides context-aware hints, where hints are generated by a separate small language model trained via distillation from a stronger LLM. While the hint-generating SLM alone is not capable of solving the problems, its collaboration with a reasoning SLM enables effective guidance, forming a cooperative two-model system for reasoning. Each hint is generated conditionally on the problem statement and the accumulated reasoning history, providing stepwise, localized guidance without revealing full solutions. This reduces error propagation and allows the reasoning model to focus on manageable subproblems. Experiments across diverse mathematical benchmarks and models demonstrate that hint assistance consistently improves reasoning accuracy for small language models, yielding substantial gains over standard prompting while preserving model efficiency. These results highlight that structured collaboration between small models—via hint generation and reasoning—offers an effective and lightweight mechanism for enhancing mathematical reasoning.

1 Introduction

Large language models (LLMs) have achieved remarkable progress in mathematical reasoning, successfully tackling problems that span elementary arithmetic, high school curricula, collegiate mathematics, competitive problem solving, and even research-level challenges (Hendrycks et al., 2021; Cobbe et al., 2021; Lewkowycz et al., 2022). A key driver of this progress is the emergence of a scalable chain-of-thought (CoT) paradigm, which allows models to generate explicit intermediate reasoning steps before producing a final answer (Wei et al., 2022). By externalizing "thinking" into token sequences, CoT provides a scalable mechanism for increasing inference-time computation and has been shown to substantially improve reasoning performance.

Nevertheless, such reasoning improvements are primarily realized by large, resource-intensive models, leaving smaller language models (SLMs)—which are often more desirable in educational and resource-constrained settings—significantly behind in multi-step mathematical reasoning (Hoffmann et al., 2022). Existing mathematical reasoning benchmarks and leaderboards are dominated by large-scale LLMs, making it an open challenge to improve the reasoning performance of SLMs without simply scaling model and data size. Prior work has explored a variety of approaches to strengthen SLM reasoning, including increasing inference-time computation via guided search (Yao et al., 2023), incorporating tool use (Gao et al., 2023; Schick et al., 2023), and applying specialized reinforcement learning techniques (Ouyang et al., 2022). However, these methods are often computationally expensive and require substantial additional resources. In practice, small models frequently struggle with long contexts and fail to maintain globally consistent reasoning, becoming confused or derailed at intermediate steps (Liu et al., 2024). This observation motivates a natural question: rather than forcing SLMs to reason end-to-end independently, can they be assisted locally to reason more effectively?

In this work, we investigate whether SLMs can solve more complex mathematical problems when provided with targeted assistance at critical reasoning steps. We frame this problem through a novel

hint-assisted mathematical reasoning (HintMR) paradigm, inspired by human learning processes in education (Chi et al., 2001). Treating the SLM as a student, we assume that it possesses partial knowledge sufficient to solve the problem, but may struggle to identify or execute the correct reasoning steps without guidance. In such cases, a well-designed hint—particularly one that focuses on the next local step rather than the full solution—can help the model better leverage its existing knowledge. To test this hypothesis, we first study the effect of oracle hints generated by frontier LLMs and verified by humans, which act as teachers capable of reliably solving the target problems. Our results show that even limited, local-step hints can substantially improve the reasoning accuracy of SLMs, revealing latent reasoning potential that is otherwise difficult to elicit.

Building on this insight, we further propose a collaborative hint-generation and reasoning framework that relies entirely on small language models. Specifically, we distill from the oracle hints to a dedicated SLM to serve as a hint generator, trained to produce targeted hints at arbitrary intermediate reasoning steps, analogous to a math tutor. The reasoning model then conditions on these hints to guide its subsequent reasoning steps. By interleaving hint generation and reasoning throughout the problem-solving process, the collaborating SLMs achieve performance that surpasses both individual models operating independently. See Figure 1 for an overview. This framework demonstrates a new form of *weak-model collaboration*, where multiple limited-capacity models jointly produce strong reasoning behavior—drawing an analogy to classic ensemble methods in machine learning, where weak classifiers can be combined to form a strong classifier (Schapire, 1990; Ji and Ma, 1996).

We evaluate our approach on four challenging mathematical reasoning benchmarks—NuminaMath (Li et al., 2024a), MATH-500 (Hendrycks et al., 2021), AIME-2024 (MAA), and AIME-2025 (Zhang and Math-AI, 2024)—across four open-source SLMs. The results show consistent and substantial performance improvements across datasets and model architectures. Notably, a single adapted hint-generation model generalizes effectively across all evaluated reasoning models, highlighting the robustness and broad applicability of the proposed hint-assisted framework for enhancing mathematical reasoning in SLMs.

In summary, our contributions are: (1) **NuminaMath-H dataset:** We construct **NuminaMath-H**, a high-quality dataset derived from NuminaMath that provides structured, step-by-step hints for mathematical reasoning, enabling both training and evaluation of hint-guided reasoning models. (2) **The HintMR framework:** We introduce HintMR, a general hint-assisted reasoning framework for SLMs that decomposes mathematical problem solving into iterative refinement steps and incorporates structured, context-aware hints during inference. (3) **Effective and efficient hint generation:** We demonstrate that LLM-generated hints (GPT-5.2) provide strong and consistent performance gains across all benchmarks, with improvements of up to +48.28 accuracy points on AIME-2024. Furthermore, we show that SLM-generated hints, obtained via distillation into DeepSeek-R1-Distill-Qwen-7B, remain highly effective—consistently outperforming no-hint baselines and in some cases matching or exceeding LLM performance—while offering a scalable and cost-efficient alternative.

2 Related Work

We briefly discuss related work here, with our full related work shown in Appendix A.

Data synthesis for math reasoning. Recent advances in mathematical reasoning are largely driven by scaling supervision via synthetic data, replacing scarce human-written solutions with large corpora of problems and step-by-step rationales. MetaMath (Yu et al., 2024) and MathInstruct (Yue et al., 2024) demonstrate that multi-view generation and structured CoT supervision improve downstream performance, while OpenMathInstruct (Toshniwal et al., 2024, 2025) scales this paradigm to millions of examples. Competition-style datasets such as NuminaMath (Li et al., 2024a,c,b) further provide diverse, challenging problems with templated solutions. Complementary efforts explore verifiability and tool use, including Big-Math (Albalak et al., 2025) and MathCoder (Wang et al., 2023). Most related to our setting, TMATH (Qi et al., 2025) focuses on problem–hint pairs, reinforcing the premise that well-scoped hints can unlock latent reasoning ability.

Mathematical reasoning with hints. Prior work shows that intermediate guidance can improve reasoning compared to fully unguided generation. Progressive-Hint Prompting (Zheng et al., 2023)

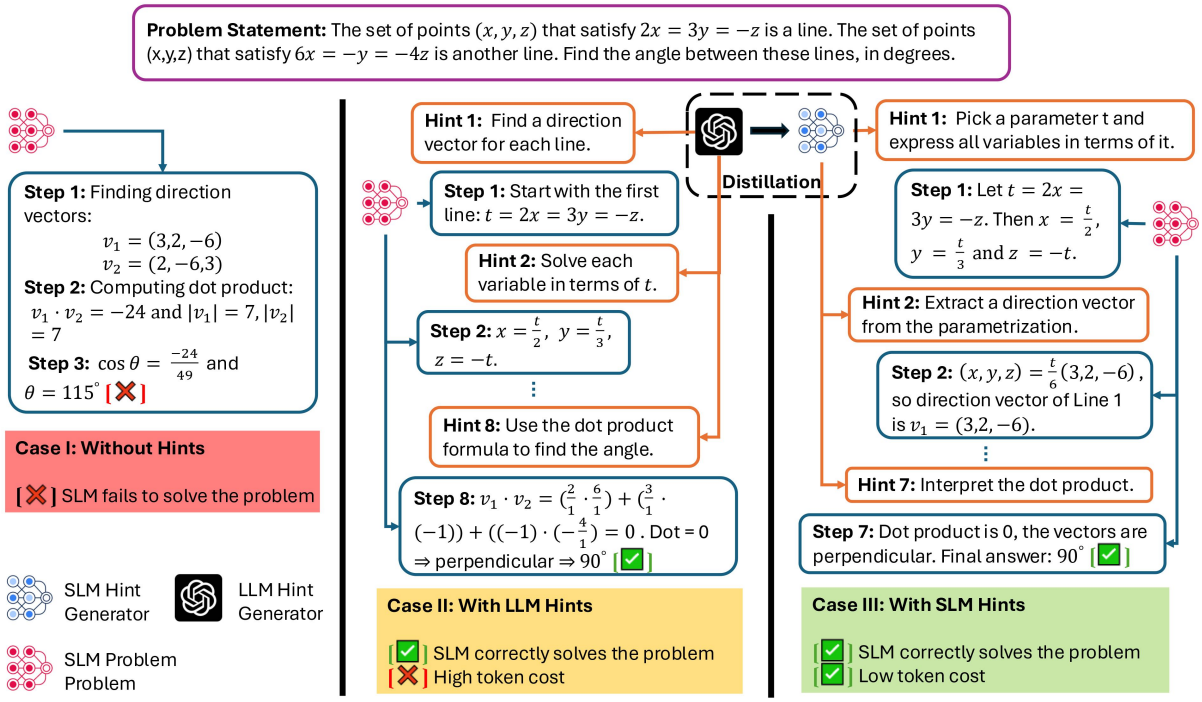

Figure 1: Overview of the HintMR inference. A LLM generates step-wise oracle hints, which are distilled into a SLM for efficient hint generation. During inference, the SLM solver incrementally constructs its reasoning trajectory using either no hints, LLM-generated hints, or SLM-generated hints. While both SLM and LLM hints enable correct solutions, SLM hints achieve comparable performance at significantly lower token cost.

and Hint-before-Solving (Fu et al., 2024) incorporate hints within single-model pipelines. In contrast, we adopt a two-SLM decomposition: a **hinter** for lightweight guidance and a **solver** for full derivations. Similar ideas appear in code reasoning, e.g., TEACH (Zhang et al., 2025), where specialized hint generators improve solution quality. This supports separating guidance from execution.

Boosting SLM performance. Recent work strengthens small models for math reasoning through continued pretraining and instruction tuning, including DeepSeekMath (Shao et al., 2024), Phi-4-mini-reasoning (Xu et al., 2025), and Qwen2.5-Math (Yang et al., 2024), as well as WizardMath (Luo et al., 2023) and MAmmoTH (Yue et al., 2023, 2024). Tool use (e.g., ToRA (Gou et al., 2023)) and inference-time scaling (e.g., rStar-Math (Guan et al., 2025)) provide complementary gains.

Our approach is also related to combining weak learners, including boosting (Freund and Schapire, 1997), distillation (Hinton et al., 2015), and process supervision (Lightman et al., 2024; Luo et al., 2024). Iterative refinement methods such as Self-Refine (Madaan et al., 2023) and Reflexion (Shinn et al., 2023) further highlight the value of stepwise feedback. Instead of explicit verification or reward modeling, we use a second SLM to provide localized, pedagogical hints conditioned on partial solutions.

3 Methodology

We present **HintMR**, a framework that distills LLM-generated knowledge into a SLM, enabling efficient generation and use of structured, step-wise hints for mathematical reasoning. The framework consists of three stages: (1) *Hint Generation*, where a strong LLM produces ordered hints; (2) *Knowledge Distillation*, where these hints are used to fine-tune a SLM via QLoRA; and (3) *Hint-Guided Reasoning*, where a solver incrementally incorporates hints to refine its reasoning and produce the final solution, as illustrated in Figure 1.

3.1 Hint Generation

The objective is to construct structured, step-wise hints that provide intermediate guidance for mathematical reasoning without revealing the final solution. To obtain high-quality supervision, we employ state-of-the-art LLM as an *oracle instructor*. For each training instance, the oracle receives the problem statement together with the corresponding ground-truth solution. Access to the complete so-

lution trajectory enables the oracle to decompose the reasoning process into a sequence of intermediate instructional hints. Given a problem P and its solution S, the LLM generates an ordered sequence of hints $H = (h_1, h_2, \ldots, h_T)$, where each hint provides guidance toward the next reasoning step while avoiding disclosure of the final answer. These hints are designed to support incremental reasoning by suggesting relevant concepts, transformations, or next actions that help progress toward the solution.

The oracle hint generation process follows three pedagogical constraints. First, each hint should guide the learner toward the next logical reasoning step. Second, hints must avoid revealing the final answer or future reasoning steps. Third, hints are phrased as instructional guidance rather than explicit solution statements. To ensure adherence to these constraints, each generated hint is carefully reviewed and refined to verify that it does not directly reveal the answer while preserving its instructional value. Using this process, we generate high-quality hint sequences for a diverse collection of mathematical reasoning problems. Each resulting training instance consists of a problem, its solution, and the corresponding ordered hint sequence (P, S, H), which forms the basis for the subsequent hint-distillation stage of the HintMR framework.

3.2 Knowledge Distillation into a SLM

After generating oracle hints, we distill the hint-generation behavior of the LLM into a SLM hint generator, as illustrated in Figure 1. The objective is to train a compact model capable of producing high-quality instructional hints without relying on a LLM during inference.

To perform this distillation, we fine-tune the SLM hint generator using three components derived from the oracle hint generation process: the problem statement P, the intermediate reasoning state W_t extracted from the ground-truth solution trajectory, and the corresponding oracle-generated hint h_t. For each problem, the reasoning states W_t are obtained by decomposing the solution trajectory into intermediate steps, and the LLM generates the corresponding instructional hint h_t for each step. Together, these elements capture how an expert model provides step-wise guidance during mathematical reasoning. The knowledge distillation is conducted via parameter-efficient fine-tuning using Quantized Low-Rank Adaptation (QLoRA). The base SLM model remains frozen throughout training, while trainable LoRA adapters are inserted into attention projection layers and feed-forward network projections. This design allows the model to adapt its behavior while updating only a small subset of parameters.

During training, each instance is serialized into a sequence consisting of the problem statement P, the intermediate reasoning state W_t, and the oracle-generated hint h_t. The model is optimized using a causal language modeling objective that maximizes the conditional likelihood of the oracle hint given the problem and the reasoning context:

$$\mathcal{L} = -\sum_{i=1}^{|h_t|} \log p_\theta(h_{t,i} \mid P, W_t, h_{t,<i}) \,,$$

where p_θ denotes the language model parameterized by θ, $h_{t,i}$ is the i-th token of the hint at step t, and $h_{t,<i}$ represents the preceding hint tokens. Through this training paradigm, the SLM learns to reproduce the instructional behavior of the LLM hint generator. Rather than learning to directly solve the problem, the distilled model learns how an expert provides pedagogically useful hints. As a result, the trained SLM hint generator can efficiently generate structured hints without requiring access to ground-truth solutions or LLMs.

3.3 Hint-Guided Reasoning

During inference, the SLM problem solver performs step-wise reasoning guided by a sequence of hints associated with the problem, as illustrated in Figure 1.

Given a problem P and an ordered sequence of hints $H = (h_1, \ldots, h_T)$, the SLM solver incrementally constructs its reasoning trajectory by incorporating one hint at a time. Rather than generating the full solution in a single pass, the SLM solver produces intermediate reasoning steps that are explicitly guided by the provided hints. Formally, the reasoning process follows a hint-conditioned update rule:

$$W_t = \begin{cases} f_\theta(P, h_1), & t = 1, \\ f_\theta(h_t, W_{t-1}), & t > 1, \end{cases}$$

where W_t denotes the intermediate reasoning state at step t, and f_θ is the SLM solver parameterized by θ. At the first step, the model conditions on both the problem and the initial hint, while subsequent steps refine the reasoning based on the previous

state and the next hint. After generating the final reasoning state W_T, the SLM solver produces the final answer $S = g_\theta(W_T)$, where g_θ denotes the solution generation function.

As illustrated in Figure 1, incorporating hints—especially those generated by the distilled SLM hint generator—enables the SLM problem solver to follow a more structured and reliable reasoning trajectory, often matching the performance of LLM-guided reasoning while maintaining significantly lower token cost. In contrast, unguided reasoning (i.e., without hints) is more prone to failure on complex problems. Overall, this step-wise hint-guided interaction provides structured guidance without constraining the solver's autonomy, reducing error accumulation in long reasoning chains and improving the reliability of SLMs on complex mathematical reasoning tasks.

4 Experimental Setup

In this section, we introduce our experimental setup including dataset, models, and evaluation. Additional experimental setup information is deferred to Appendix B.

4.1 Datasets

NuminaMath-H: A human-curated math reasoning hints dataset. We create NuminaMath-H, a human-curated hint-augmented mathematical reasoning dataset, constructed from the NuminaMath [1] benchmark. NuminaMath-H pairs each mathematical problem with an ordered sequence of instructional hints designed to guide step-wise reasoning without revealing the final answer. To construct the hint supervision, we generated hints for two separate subsets of NuminaMath serving different purposes. First, we used the first 75 International Mathematical Olympiad (IMO)-style problems from NuminaMath to create the NuminaMath-H evaluation set, which is used exclusively for testing. Later, we selected another 101 IMO-style problems from NuminaMath to construct the supervision used for training the hint generator SLM (Section 3.2). These two subsets are disjoint, ensuring that the problems used for training do not overlap with the evaluation set.

The original NuminaMath dataset provides both problem statements and full reference solutions. We therefore generated initial hints using GPT-5.2 by conditioning on the problem statement and the corresponding solution, prompting the model to produce step-wise instructional guidance. All generated hints were subsequently *manually checked and rewritten by human experts* to ensure mathematical correctness, clarity, and pedagogical alignment with the intended reasoning process. Across both the training and evaluation subsets, the average number of hints per problem is 8.

Other datasets. Besides NuminaMath, we also construct hint-augmented datasets for MATH-500 [2], AIME-2024 [3], and AIME-2025 [4]. We follow the same hint-generation procedure used for NuminaMath. From MATH-500, we randomly sampled 74 problems, producing an average of 10 hints per problem. For AIME-2024, we used 29 problems out of the original 30 instances, excluding one problem due to consistently ambiguous hint generation identified during verification with GPT-5.2; the resulting dataset contains an average of 12 hints per problem. The AIME-2025 dataset contains 15 problems, all of which were included. Since AIME-2025 provides only final answers without full solutions, GPT-5.2 was conditioned on the problem statement and the final answer to generate instructional hints. The average number of hints for AIME-2025 is 14.

4.2 Models and Evaluation

We evaluate our framework on four compact, distilled SLM problem solvers: *Qwen2.5-Math-7B-Instruct* (Yang et al., 2024), *DeepSeek-R1-Distill-Qwen-7B* and *DeepSeek-R1-Distill-Llama-8B* (Guo et al., 2025), and *Phi-4-mini-reasoning* (Xu et al., 2025). In addition, we evaluate **FT DeepSeek-R1-Distill-Qwen-7B**, a fine-tuned SLM hint generator, to examine whether it can directly solve problems beyond its primary role of producing instructional hints. All models are used in inference-only mode without additional task-specific fine-tuning, and 4-bit quantization is applied when supported to ensure efficient execution.

During inference, SLM solvers receive ordered hint sequences generated by the SLM hint generator, which are incorporated step-by-step to guide intermediate reasoning updates. We compare

[1] https://huggingface.co/datasets/AI-MO/NuminaMath-CoT

[2] https://huggingface.co/datasets/HuggingFaceH4/MATH-500

[3] https://huggingface.co/datasets/Maxwell-Jia/AIME_2024

[4] https://huggingface.co/datasets/opencompass/AIME2025

this *hint-assisted* setting against a *no-hint baseline*, where models generate solutions directly as $S_{\text{no-hint}} = g_\theta(P)$ without intermediate refinement.

For evaluation, each generated solution is assessed via a joint human and GPT-5.2 verification process to determine mathematical correctness. Accuracy is computed as a binary outcome, problem solved or not, based on this validation, with ambiguous cases resolved through additional human review to ensure reliability.

5 Results and Analysis

In this section, we present our main results including reasoning with LLM hints, fined-tuned SLM hints, non-fine-tuned SLM hints, and efficiency analysis. Our performance stability analysis is deferred to Appendix C. In Appendix D we show three qualitative examples of math reasoning without hints, with LLM hints, and with SLM hints.

5.1 Reasoning with LLM Hints

Table 1 reports accuracy comparisons for SLMs with and without hint assistance across four mathematical reasoning benchmarks of increasing difficulty. Across all datasets, with LLM hints or fine-tuned SLM hints, SLMs consistently outperform the no-hint baseline, with larger gains generally observed on more complex tasks and for higher-capacity models.

On NuminaMath, hint guidance yields substantial improvements across all models. Qwen2.5-Math-7B-Instruct benefits most from LLM hints, reaching 78.67% accuracy (+32.00), while DeepSeek-R1-Distill-Qwen-7B shows stronger relative gains from SLM hints (+16.00). On MATH-500, improvements are smaller but consistent, with Qwen2.5-Math-7B-Instruct achieving the highest accuracy (85.14%) and DeepSeek-R1-Distill-Qwen-7B attaining its best performance with SLM hints, suggesting that model-aligned compact guidance can rival LLM hints on mid-difficulty problems.

The impact of hint assistance is most pronounced on AIME-2024, where baseline performance is extremely low. LLM hints lead to dramatic gains, particularly for DeepSeek-R1-Distill-Qwen-7B (68.97%, +48.28) and DeepSeek-R1-Distill-Llama-8B (from 0.00% to 44.83%). While SLM hints also improve performance, they consistently trail LLM hints on this benchmark, highlighting the value of richer guidance for long-horizon reasoning. On AIME-2025, overall accuracy remains low due to increased difficulty and limited sample size, but selective gains from hint assistance are still observed, indicating that hints can partially mitigate capacity limitations in extreme settings.

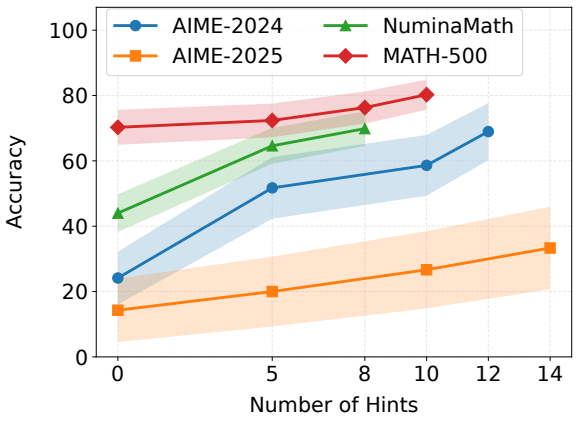

Figure 2: Performance curves showing improvement in reasoning quality with more hints.

In addition, as illustrated in Figure 2, we show that hint guidance enables the model to progressively refine its reasoning trajectory, leading to substantial improvements in final answer accuracy compared to the no-hint setting.

5.2 Reasoning with Fine-Tuned SLM Hints

Across all datasets, SLM-generated hints consistently improve performance over the no-hint baseline, demonstrating that compact models can provide useful stepwise guidance even without access to a large teacher model at inference time. On NuminaMath, SLM hints yield accuracy gains ranging from +2.66 to +16.00 points, with particularly strong improvements for DeepSeek-R1-Distill-Qwen-7B (+16.00) and Phi-4-mini-reasoning (+8.00). Similar trends are observed on MATH-500, where SLM hints improve accuracy for all evaluated models, achieving gains of up to +20.28 points and, in one case, matching or slightly exceeding the performance obtained with LLM-generated hints. These results indicate that SLM hints can effectively scaffold multi-step reasoning and reduce error accumulation, even when generated by models with limited capacity.

While LLM-generated hints generally provide larger gains, SLM hints remain competitive and often capture a substantial fraction of the improvement. On MATH-500, DeepSeek-R1-Distill-Qwen-7B achieves 85.14% accuracy with SLM hints, outperforming its LLM-hint counterpart, and other

Dataset	Model	w/o Hints	w/ LLM Hints	w/ SLM Hints	w/ NFT SLM Hints
NuminaMath	Qwen2.5-Math-7B-Instruct	46.67% (35/75)	**78.67%** (59/75)	58.67% (44/75)	36.00% (27/75)
	DeepSeek-R1-Distill-Qwen-7B	44.00% (33/75)	69.33% (52/75)	**60.00%** (45/75)	42.67% (32/75)
	DeepSeek-R1-Distill-Llama-8B	42.67% (32/75)	68.00% (51/75)	45.33% (34/75)	26.67% (20/75)
	Phi-4-mini-reasoning	21.33% (16/75)	33.33% (25/75)	29.33% (22/75)	21.33% (16/75)
	FT DeepSeek-R1-Distill-Qwen-7B	38.67% (29/75)	65.33% (49/75)	37.33% (28/75)	**45.33%** (34/75)
MATH-500	Qwen2.5-Math-7B-Instruct	75.68% (56/74)	**85.14%** (63/74)	79.73% (59/74)	54.05% (40/74)
	DeepSeek-R1-Distill-Qwen-7B	64.86% (48/74)	82.43% (61/74)	**85.14%** (63/74)	**68.92%** (51/74)
	DeepSeek-R1-Distill-Llama-8B	52.70% (39/74)	71.62% (53/74)	68.92% (51/74)	52.70% (39/74)
	Phi-4-mini-reasoning	56.76% (42/74)	77.03% (57/74)	74.32% (55/74)	58.01% (43/74)
	FT DeepSeek-R1-Distill-Qwen-7B	60.81% (45/74)	64.86% (48/74)	64.86% (48/74)	68.92% (51/74)
AIME-2024	Qwen2.5-Math-7B-Instruct	13.79% (4/29)	41.38% (12/29)	20.69% (6/29)	10.34% (3/29)
	DeepSeek-R1-Distill-Qwen-7B	20.69% (6/29)	**68.97%** (20/29)	34.48% (10/29)	**27.59%** (8/29)
	DeepSeek-R1-Distill-Llama-8B	0.00% (0/29)	44.83% (13/29)	20.69% (6/29)	17.24% (5/29)
	Phi-4-mini-reasoning	6.90% (2/29)	26.09% (6/29)	13.69% (4/29)	0.00% (0/29)
	FT DeepSeek-R1-Distill-Qwen-7B	17.24% (5/29)	44.83% (13/29)	31.03% (9/29)	24.14% (7/29)
AIME-2025	Qwen2.5-Math-7B-Instruct	13.33% (2/15)	13.33% (2/15)	13.33% (2/15)	**26.67%** (4/15)
	DeepSeek-R1-Distill-Qwen-7B	13.33% (2/15)	**20.00%** (3/15)	**20.00%** (3/15)	6.67% (1/15)
	DeepSeek-R1-Distill-Llama-8B	6.67% (1/15)	6.67% (1/15)	6.67% (1/15)	6.67% (1/15)
	Phi-4-mini-reasoning	0.00% (0/15)	13.33% (2/15)	6.67% (1/15)	0.00% (0/15)
	FT DeepSeek-R1-Distill-Qwen-7B	13.33% (2/15)	13.33% (2/15)	0.00% (0/15)	0.00% (0/15)

Table 1: Accuracy comparison (%) of small language models with and without hint assistance across four mathematical reasoning datasets. Best with-hints performance per dataset is bolded. Accuracy is computed based on human and GPT-5.2 verified correctness.

models show only modest gaps between SLM and LLM hinting. On AIME-2024, although the absolute performance remains lower due to the difficulty of the benchmark, SLM hints still provide meaningful improvements across all models, with gains up to +20.69 points, demonstrating that even partial or noisier guidance can be beneficial in challenging mathematical reasoning settings. Notably, the fine-tuned hint generator model (FT DeepSeek-R1-Distill-Qwen-7B) achieves a strong accuracy of 65.52% on AIME-2024 when paired with FT SLM hints, substantially outperforming its own no-hint baseline and highlighting that the reasoning patterns learned during hint-generation training can also support effective problem solving. These findings suggest that the effectiveness of hint assistance depends not only on hint quality but also on model alignment and task structure.

On the most challenging and low-sample benchmark, AIME-2025, gains from both LLM and SLM hints are more limited, reflecting the intrinsic difficulty and variance of the task. Nevertheless, SLM hints never degrade performance and in several cases match the improvements achieved by LLM hints, indicating robustness despite reduced supervision quality. Overall, these results highlight a favorable trade-off between performance and resource efficiency: SLM-generated hints provide a lightweight yet effective alternative to LLM-based hinting, enabling scalable deployment of hint-assisted reasoning in resource-constrained settings without requiring large models at inference.

Overall, these results show that hint-assisted reasoning substantially enhances mathematical problem solving in small language models, with LLM hints providing the largest benefits on the hardest benchmarks, and SLM hints offering competitive—and sometimes superior—performance on moderately difficult tasks.

5.3 Reasoning with Non-Fine-Tuned SLM Hints

Table 1 shows that non-fine-tuned SLM (NFT-SLM) hints exhibit large variance in effectiveness across both datasets and models. On NuminaMath, NFT-SLM performance is generally low, dropping to 36.00% for Qwen2.5-Math-7B-Instruct and 26.67% for DeepSeek-R1-Distill-Llama-8B—often below the no-hint baseline—indicating that unaligned hints can actively hinder structured reasoning. Interestingly, the fine-tuned hint generator model (FT DeepSeek-R1-Distill-Qwen-7B) achieves 45.33% accuracy (34/75) on NuminaMath when paired with NFT-SLM hints, outperforming several solver models under the same setting. In contrast, on MATH-500, NFT-SLM hints reach comparatively higher accuracies (up to 68.92% with DeepSeek-R1-Distill-

Dataset	Method	Avg Time (s)	Avg Tokens	Acc. (%)
MATH-500	No-Hint	61.35	920	64.86
	HintMR	165.34	1180	**82.43**
	SC (K=8)	689.46	7350	70.27
AIME-2024	No-Hint	88.84	929	20.69
	HintMR	205.43	781	**68.97**
	SC (K=8)	805.33	7670	20.69
AIME-2025	No-Hint	102.38	901	13.33
	HintMR	241.83	872	**20.00**
	SC (K=8)	819.96	7515	13.33
NuminaMath	No-Hint	74.33	735	44.00
	HintMR	248.27	627	**69.33**
	SC (K=8)	689.46	6350	48.00

Table 2: Accuracy–compute trade-off for DeepSeek-R1-Distill-Qwen-7B across datasets. HintMR achieves significantly higher accuracy than no-hint inference with moderate computational overhead, while remaining substantially more efficient than SC decoding.

Qwen-7B), occasionally outperforming no-hint inference and suggesting some benefit for longer, procedural problems.

For AIME-2024, NFT-SLM accuracy ranges from 0.00% to 27.59%, providing measurable but limited gains over no hints for stronger models, while remaining far below LLM-hint performance. Results on AIME-2025 are mixed and noisy due to the smaller evaluation set, with accuracies spanning 0.00%–26.67%. Overall, these patterns highlight that NFT-SLM hints provide coarse and unstable guidance, offering occasional improvements on difficult benchmarks but failing to deliver consistent or robust reasoning gains.

5.4 Efficiency Analysis

Computational efficiency. To evaluate the computational efficiency of hint-assisted reasoning, we compare HintMR against no-hint inference and self-Consistency (SC) baseline (K=8), where K denotes the number of sampled reasoning trajectories used for majority voting, using DeepSeek-R1-Distill-Qwen-7B. Table 2 reports the average inference time, token usage, and final accuracy across datasets.

HintMR consistently achieves a strong balance between accuracy and computational cost. Compared to no-hint inference, HintMR substantially improves accuracy with moderate increases in inference time and token usage. In contrast, self-consistency incurs significantly higher computational cost—up to 6–8× more tokens and time—while often providing limited or no accuracy gains. On challenging benchmarks such as AIME-2024, HintMR improves accuracy from 20.69% to 68.97%, whereas self-consistency fails to improve performance despite substantially higher compute cost. These results highlight that structured hint guidance provides a more efficient alternative to brute-force sampling strategies for enhancing reasoning in SLMs.

Dataset	LLM Tokens	SLM Tokens	Reduction (%)
NuminaMath	198.43	145.36	26.74%
MATH-500	212.58	149.82	29.53%
AIME-2024	236.79	152.70	35.52%
AIME-2025	229.66	148.91	35.18%

Table 3: Average tokens per problem for hint generation. SLM hints consistently require fewer tokens than LLM hints, reducing cost and latency.

Token efficiency of SLM vs. LLM hint generation. We compare the token efficiency of SLM- and LLM-generated hints across four mathematical reasoning datasets. We use GPT-5.2 as the LLM hinter and FT DeepSeek-R1-Distill-Qwen-7B as the SLM hinter. Table 3 reports the average number of tokens per problem. SLM-generated hints consistently use fewer tokens, yielding lower computational cost and faster inference.

6 Conclusion

This work presents a hint-assisted mathematical reasoning framework for SLMs that improves multi-step problem solving through structured, stepwise guidance. By decomposing solutions into intermediate reasoning steps and providing context-aware hints, the approach mitigates error accumulation and enables models to focus on local subproblems while maintaining coherence with the overall solution trajectory. Unlike methods that rely on full-chain supervision or large-scale model capacity, our framework offers a lightweight and flexible mechanism for enhancing reasoning performance. Empirical results across multiple benchmarks demonstrate consistent accuracy improvements, highlighting the effectiveness of hint assistance as a practical tool for boosting mathematical reasoning in SLMs.

More broadly, our findings suggest that separating guidance from execution is a promising direction for scalable reasoning. This perspective naturally extends to multi-agent settings, where specialized components (e.g., hinters and solvers) collaborate, as well as to self-improving systems that iteratively refine their own reasoning through generated feedback.

References

Alon Albalak, Duy Phung, Nathan Lile, Rafael Rafailov, Kanishk Gandhi, Louis Castricato, Anikait Singh, Chase Blagden, Violet Xiang, Dakota Mahan, and Nick Haber. 2025. Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning converts weak language models to strong language models. In *International Conference on Machine Learning*, pages 6621–6642. PMLR.

Michelene TH Chi, Stephanie A Siler, Heisawn Jeong, Takashi Yamauchi, and Robert G Hausmann. 2001. Learning from human tutoring. *Cognitive science*, 25(4):471–533.

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

Yoav Freund and Robert E. Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*.

Jinlan Fu, Shenzhen Huangfu, Hang Yan, See-Kiong Ng, and Xipeng Qiu. 2024. Hint-before-solving prompting: Guiding llms to effectively utilize encoded knowledge.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International conference on machine learning*, pages 10764–10799. PMLR.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Tora: A tool-integrated reasoning agent for mathematical problem solving.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. 2025. rStar-math: Small LLMs can master math reasoning with self-evolved deep thinking. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 20640–20661. PMLR.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, and 1 others. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Chuanyi Ji and Sheng Ma. 1996. Combinations of weak classifiers. *Advances in Neural Information Processing Systems*, 9.

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

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. 2024a. Numinamath.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. 2024b. Numinamath.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. 2024c. Numinamath tir.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*. ArXiv:2305.20050.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, Yansong Tang, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. 2024. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*.

MAA. American invitational mathematics examination-aime 2024, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback.

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.

Changyong Qi, Yuang Wei, Haoxin Xu, Longwei Zheng, Peiji Chen, and Xiaoqing Gu. 2025. TMATH: A dataset for evaluating large language models in generating educational hints for math word problems. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5082–5093.

Robert E Schapire. 1990. The strength of weak learnability. *Machine learning*, 5(2):197–227.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning.

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. 2025. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. In *International Conference on Learning Representations (ICLR)*. ArXiv:2410.01560.

Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. 2024. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. *Advances in Neural Information Processing Systems*, 37:34737–34774.

Ke Wang, Houxing Ren, Aojun Zhou, Zimu Lu, Sichun Luo, Weikang Shi, Renrui Zhang, Linqi Song, Mingjie Zhan, and Hongsheng Li. 2023. Mathcoder: Seamless code integration in llms for enhanced mathematical reasoning.

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

Haoran Xu, Baolin Peng, Hany Awadalla, Dongdong Chen, Yen-Chun Chen, Mei Gao, Young Jin Kim, Yunsheng Li, Liliang Ren, Yelong Shen, Shuohang Wang, Weijian Wu, Jianfeng Gao, and Weizhu Chen. 2025. Phi-4-mini-reasoning: Exploring the limits of small reasoning language models in math. *arXiv preprint arXiv:2504.21233*.

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.

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

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap your own mathematical questions for large language models. In *International Conference on Learning Representations (ICLR)*. ArXiv:2309.12284.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning.

Xiang Yue, Tianyu Zheng, Ge Zhang, and Wenhu Chen. 2024. Mammoth2: Scaling instructions from the web. *Advances in Neural Information Processing Systems*, 37:90629–90660.

Yating Zhang, Wei Dong, Jiaxin Liu, Shangwen Wang, Deze Wang, Tiecheng Ma, Yiwei Li, and Kang Yang. 2025. A little help goes a long way: Tutoring llms in solving competitive programming through hints. *IEEE Transactions on Software Engineering*.

Yifan Zhang and Team Math-AI. 2024. American invitational mathematics examination (aime) 2025. *Wei Zhao, Zhe Li, Yige Li, Ye Zhang, and Junfeng Sun*.

Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. 2023. Progressive-hint prompting improves reasoning in large language models.

A Related Work

Data synthesis for math reasoning. A key driver behind recent gains in mathematical reasoning is the ability to scale supervision through synthetic data: instead of relying on scarce human-written solutions, many pipelines now bootstrap large corpora of problems paired with step-by-step rationales. MetaMath (Yu et al., 2024) exemplifies this trend by rewriting seed problems from multiple perspectives to construct MetaMathQA, showing that multi-view generation can substantially improve downstream CoT fine-tuning even at the 7B scale. In a similar spirit, MathInstruct (Yue et al., 2024) emphasizes step-by-step solutions explicitly designed to elicit CoT behaviors during fine-tuning. OpenMathInstruct (Toshniwal et al., 2024, 2025) pushes this direction further, scaling to millions (and then tens of millions) of problem–solution pairs and analyzing how choices such as solution format, teacher strength, and diversity affect performance. In parallel, competition-style collections such as NuminaMath (Li et al., 2024a,c,b) provide broad coverage of contest problems and templated step-by-step solutions, offering a strong substrate for training and evaluating small models on challenging multi-step tasks.

Other efforts target complementary axes, including RL-compatibility and verifiability of targets (e.g., Big-Math (Albalak et al., 2025)) and interleaving natural language with executable code and tool traces (e.g., MathCoder/MathCodeInstruct (Wang et al., 2023)), which can stabilize long computations. Most closely aligned with our goal of localized assistance is TMATH (Qi et al., 2025), which focuses on pedagogical problem–hint pairs and evaluates models on generating hints for math word problems; while TMATH is designed for helping human learners, it reinforces the premise we study here: well-scoped hints can unlock latent competence, and hint generation deserves dedicated data and evaluation.

Mathematical reasoning with hints. A growing body of work suggests that models can be guided more effectively when they are not forced to reason end-to-end in isolation, but instead receive intermediate guidance that nudges them back onto a productive trajectory. Progressive-Hint Prompting (Zheng et al., 2023) uses prior attempts as hints in a multi-turn interaction, while Hint-before-Solving Prompting (Fu et al., 2024) asks the model to first generate high-level hints before producing solutions. Compared with these predominantly single-model or hint-first approaches, our method employs a two-SLM decomposition with explicit specialization: a **hinter** that generates lightweight guidance and a **solver** that produces step-by-step derivations. A closely related line appears in code reasoning: Zhang et al. (2025) proposed the TEACH paradigm that tutors LLMs for competitive programming by introducing a domain-specialized hint generator trained on curated platform hints, and shows that concise, targeted algorithmic hints can improve downstream solution generation. This strengthens our motivation for separating guidance from execution: a hinter can focus on surfacing the key idea (e.g., the intended algorithmic trick or mathematical lemma), while a solver focuses on producing a complete, verifiable derivation.

Boosting SLM performance. The gap between frontier models and SLMs has also motivated a wave of work aimed at strengthening open-weight, modest-parameter systems for math reasoning. DeepSeekMath (Shao et al., 2024) continues pretraining on math-heavy corpora and introduces RL-style post-training to strengthen 7B models. The Phi-4-mini-reasoning (Xu et al., 2025) is another compact 3.8B-model specifically optimized for advanced mathematical reasoning and step-by-step problem-solving. Qwen2.5-Math (Yang et al., 2024) proposes a self-improvement pipeline and releases math-specialized models down to 1.5B parameters. These models are the building blocks for our proposed approach. Instruction-tuned math models further show that curated rationales and instruction construction can close gaps for small models, including WizardMath (Luo et al., 2023) and MAmmoTH/MAmmoTH2 (Yue et al., 2023, 2024). Tool-integrated agents such as ToRA (Gou et al., 2023) and code-interleaved training such as MathCoder (Wang et al., 2023) highlight complementary gains from interacting with external computation. Finally, recent test-time scaling and search methods demonstrate that small models can benefit substantially from additional inference-time compute, e.g., rStar-Math (Guan et al., 2025).

Our work is also connected to broader ideas in combining weak learners into stronger systems, from classical boosting (Freund and Schapire, 1997) to knowledge distillation (Hinton et al., 2015). A complementary direction provides step-level feedback signals: process supervision and verifiers (Lightman et al., 2024) show advantages over outcome-only supervision for math reasoning,

and automated variants scale step-level supervision without human annotation (Luo et al., 2024). Iterative self-improvement methods further improve outputs using feedback loops, including natural-language feed back during inference (e.g., Self-Refine (Madaan et al., 2023) and Reflexion (Shinn et al., 2023)) and synthetic feedback during training (e.g. Self-Play Fine-Tuning (SPIN) (Chen et al., 2024)). Our approach shares the same intuition (local, stepwise guidance is valuable), but replaces explicit verification/reward modeling with a second SLM trained or prompted to provide pedagogical hints conditioned on the solver's partial work.

B Additional Experimental Setup Details

B.1 Computational Environment

All experiments were conducted using Google Colab Pro with an NVIDIA L4 GPU. The runtime environment provided approximately 53.0 GB of system RAM, 22.5 GB of GPU memory, and 235.7 GB of local disk storage. This configuration enabled efficient inference for all evaluated models while maintaining consistent experimental conditions across datasets.

B.2 Models

We evaluate our framework on four compact and distilled language models suitable for resource-constrained mathematical reasoning: Qwen2.5-Math-7B-Instruct (Yang et al., 2024), DeepSeek-R1-Distill-Qwen-7B and DeepSeek-R1-Distill-Llama-8B (Guo et al., 2025), and Phi-4-mini-reasoning (Xu et al., 2025). In addition to these four models, we also evaluate **FT DeepSeek-R1-Distill-Qwen-7B**, which is the fine-tuned hint generator model. This allows us to examine whether the hint generator itself can solve the problems directly, beyond its primary role of producing instructional hints. All models are used in inference-only mode without additional fine-tuning, and 4-bit quantization is applied when supported to enable efficient execution. During inference, the solver models are provided with the ordered hint sequences generated by the hint generator SLM described in Section 3.2. These hints are incorporated after each reasoning step to guide the solver toward the next step of the solution. We compare hint-assisted reasoning against a no-hint baseline that directly generates a solution $S_{\text{no-hint}} = g_\theta(P)$ without intermediate refinement. In our implementation, refinement steps use deterministic decoding, while final solution synthesis employs mildly stochastic decoding under a constrained JSON output format. The hint generator SLM is fine-tuned using parameter-efficient QLoRA. Specifically, we fine-tune DeepSeek-R1-Distill-Qwen-7B to learn the oracle hint-generation behavior. Fine-tuning is performed for two epochs with an effective batch size of 8 and a learning rate of 1×10^{-4}. The model is optimized using cosine learning-rate scheduling with warmup and early stopping based on validation loss. A concise summary of the fine-tuning configuration is provided in Table 4.

Hyperparameter	**Value**
LoRA Rank (r)	16
LoRA Scaling (α)	32
LoRA Dropout	0.1
Optimizer	8-bit Paged AdamW
Learning Rate	1×10^{-4}
Learning Rate Scheduler	Cosine
Warmup Ratio	0.03
Epochs	2
Max Sequence Length	1536
Train Batch Size (per device)	1
Gradient Accumulation Steps	8
Effective Batch Size	8
Gradient Checkpointing	Enabled
Evaluation Strategy	Step-based
Evaluation Frequency	Every 10 steps
Early Stopping Patience	3 evaluations
Metric for Model Selection	Validation Loss

Table 4: Hyperparameter settings for fine-tuning DeepSeek-R1-Distill-Qwen-7B.

B.3 Evaluations

For evaluation, we compare two inference settings: a *hint-assisted* setting, in which models receive the full ordered sequence of hints during inference, and a *no-hint baseline*, in which models generate solutions directly from the problem statement without intermediate guidance. Each generated solution is independently evaluated by human annotators and GPT-5.2, which jointly determine whether the predicted solution is mathematically correct. Accuracy is computed based on this binary judgment, with a solution marked correct only when it is verified as valid and incorrect otherwise; ambiguous cases are resolved through additional human review to ensure evaluation reliability.

C Additional Experimental Results

C.1 Statistical Stability Across Runs

To account for stochasticity in model generation, we repeat each experiment 8 times with different random seeds and report the mean performance along with the standard error. Given the relatively small size of the AIME-2024 and AIME-2025 datasets, repeated runs help obtain a more stable and reliable estimate of performance. Let $\{x_1, \ldots, x_N\}$ denote the accuracy across $N = 8$ runs. Table 5 reports the mean accuracy $\pm$ SEM for **FT DeepSeek-R1-Distill-Qwen-7B**. LLM hints provide substantial gains on AIME-2024, while SLM hints also improve over the baseline; however, all methods struggle on AIME-2025, and the low standard errors indicate stable results.

Dataset	Setting	Mean Acc. (%)
AIME-2024	w/o Hints	18.10 ± 0.86
	w/ LLM Hints	46.55 ± 1.59
	w/ SLM Hints	29.31 ± 1.13
	w/ NFT SLM Hints	23.28 ± 0.86
AIME-2025	w/o Hints	9.17 ± 1.22
	w/ LLM Hints	8.34 ± 1.09
	w/ SLM Hints	0.00 ± 0.00
	w/ NFT SLM Hints	0.00 ± 0.00

Table 5: Performance stability across 8 runs for **FT DeepSeek-R1-Distill-Qwen-7B** on AIME-2024 and AIME-2025 datasets.

D Qualitative Example Analysis

Here we provide a qualitative example analysis with results shown in Figure 3-5. The example shows a clear progression in reasoning quality as structured guidance is introduced.

Without hints, the model performs locally correct derivations but fails to incorporate the global constraint required to identify the correct interior solution, causing the reasoning process to stall before reaching the final answer. Introducing hints significantly improves the reasoning trajectory by guiding the model towards the correct algebraic pathway and helping it operationalize the global structure of the problem. Even with SLM-generated hints, the model's reasoning becomes more structured and goal-directed, enabling it to progressively correct its reasoning path and ultimately recover the correct answer, although some intermediate steps remain implicit. LLM-generated hints further strengthen this process by explicitly encoding the problem's geometric and constraint structure, producing a clearer and more faithful step-by-step reasoning trajectory. Overall, these results demonstrate that hint-assisted reasoning substantially improves model reasoning quality, and that even lightweight SLM hints can meaningfully guide the model toward correct solutions.

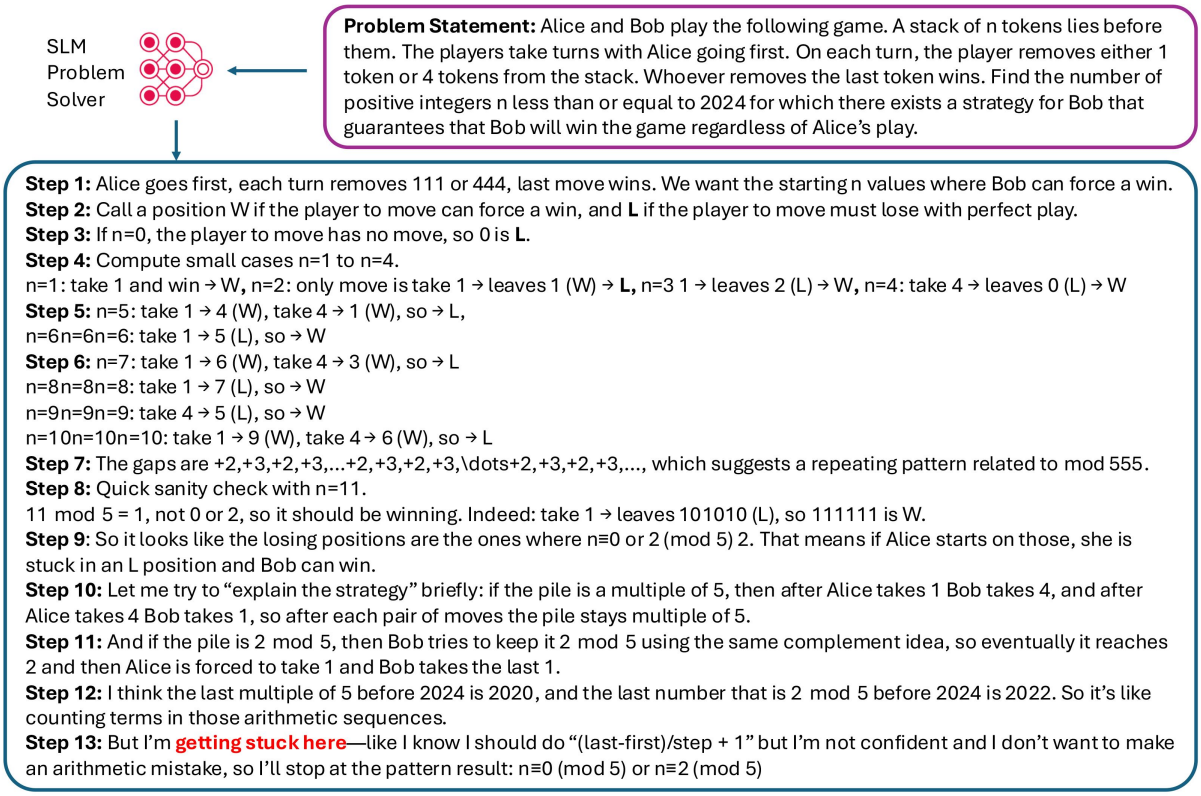

Figure 3: Summarized reasoning trajectory of a predicted solution generated by DeepSeek-R1-Distill-Qwen-7B without hints. Without external guidance, the model struggles to maintain a coherent reasoning path and fails to reach the correct final answer, repeatedly producing phrases such as "getting stuck here" (highlighted in red), indicating stalled reasoning.

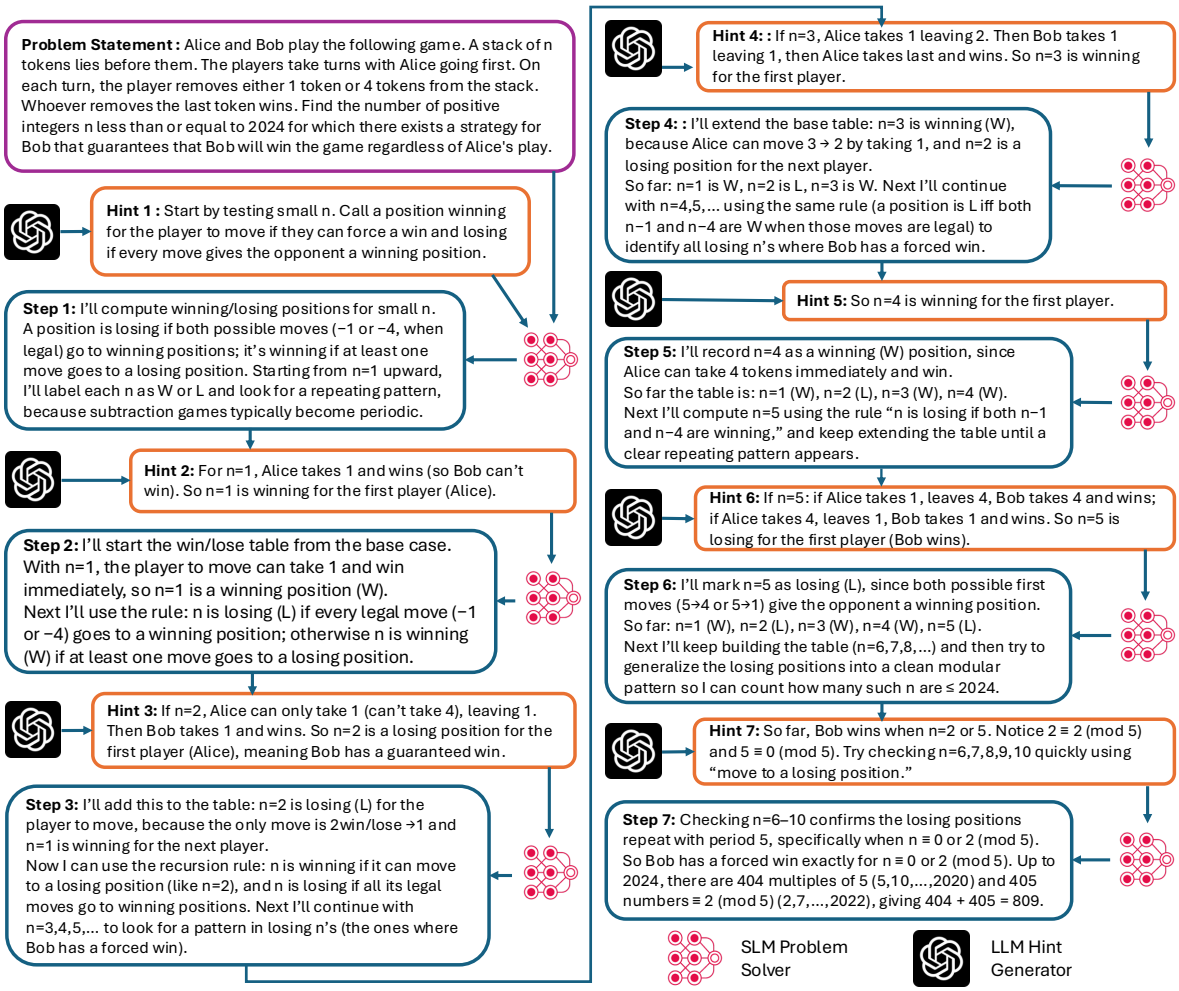

Figure 4: Summarized reasoning trajectory of a predicted solution generated by DeepSeek-R1-Distill-Qwen-7B using LLM-generated hints. With guidance from 7 hints, the model maintains a coherent reasoning path, progressively identifying the pattern in the game states and ultimately arriving at the correct final answer.

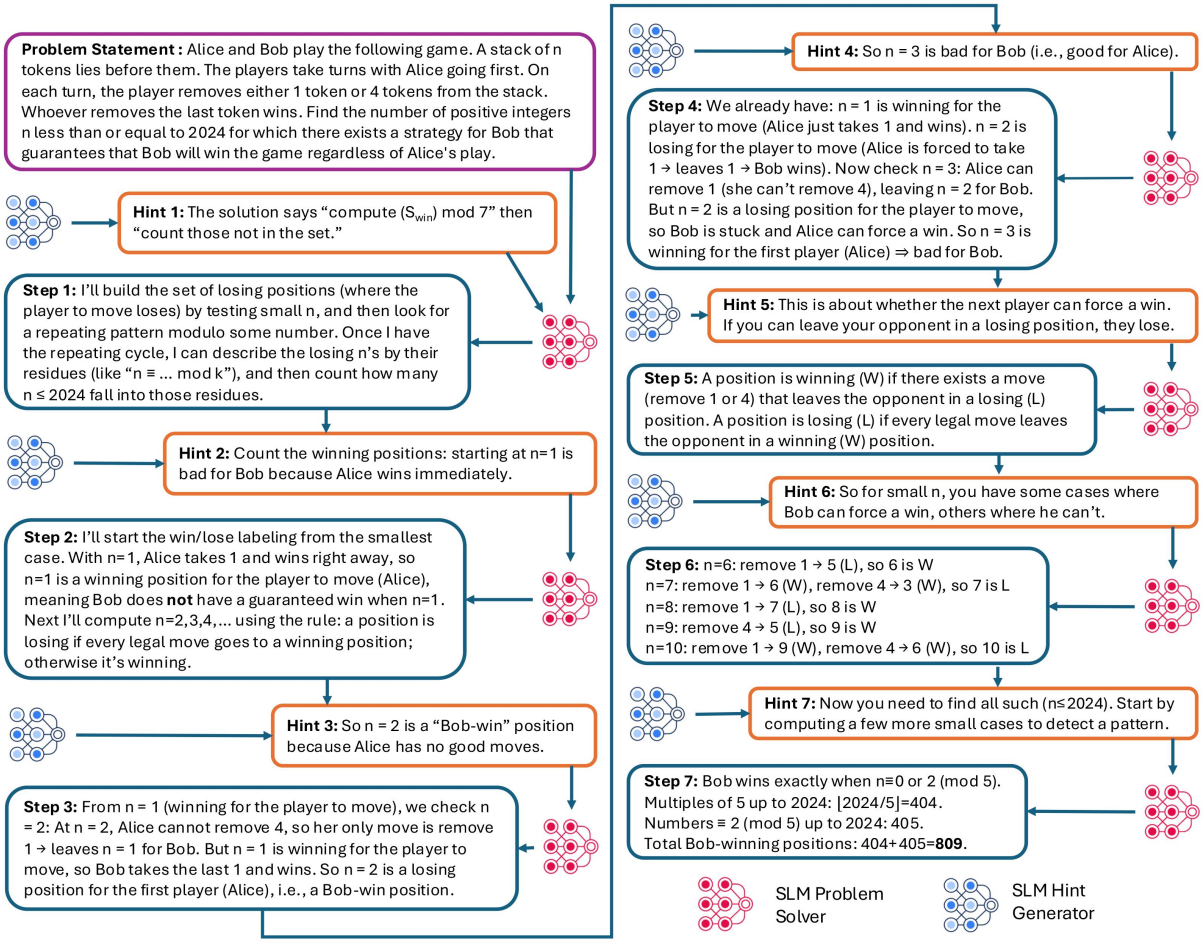

Figure 5: Summarized reasoning trajectory of a predicted solution generated by DeepSeek-R1-Distill-Qwen-7B using SLM-generated hints. Despite the first hint being suboptimal, the subsequent hints guide the model toward a coherent reasoning path, allowing it to reach the correct final answer using 7 hints, the same number required when using LLM-generated hints.