Learning to Refine: Self-Refinement of Parallel Reasoning in LLMs

Qibin Wang[1,2], Pu Zhao[1*], Shaohan Huang[1], Fangkai Yang[1], Lu Wang[1], Furu Wei[1], Qingwei Lin[1], Saravan Rajmohan[1], Dongmei Zhang[1]

[1]Microsoft, [2]Peking University

wangqibin@stu.pku.edu.cn, {Pu.Zhao, shaohanh, fangkai.yang, wlu, fuwei, qlin, sarr1, dongmeiz}@microsoft.com

Abstract

Test-time scaling (TTS) has gained widespread attention for enhancing LLM reasoning. Existing approaches such as Best-of-N and majority voting are limited as their performance depends on the quality of candidate responses, making them unable to produce a correct solution when all candidates are incorrect. Parallel self-refinement, generating multiple candidates and synthesizing a refined answer conditioned on them, offers a promising alternative, but the underlying mechanism driving its effectiveness remains obscure. To bridge this gap in understanding, we introduce a new metric, the Refinement Gap, designed to quantify the relative improvement of self-refinement beyond majority voting. We show that the Refinement Gap exhibits a clear scaling trend with model size and is only weakly correlated with the base capability. Based on this discovery, we propose Generative Self-Refinement (GSR), a parallel test-time scaling framework that transfers the refinement policy from larger teacher models with higher refinement gap into smaller students. Crucially, GSR jointly trains a single model to generate strong candidates and refine a better final answer based on these candidates. Experimental results demonstrate that our method achieves state-of-the-art performance across five mathematical benchmarks over other parallel aggregation methods, while the learned refinement skill transfers across multiple model scales and families and exhibits robust generalization to an out-of-distribution domain.

1 Introduction

Test-time scaling (TTS) (Brown et al., 2024; Snell et al., 2024) has become a dominant strategy, in which model performance can be consistently improved by allocating additional compute at inference time. A common strategy is majority voting (Wang et al., 2023), which leverages the principle of self consistency to improve performance, identifying the most frequent answer from multiple reasoning paths. Best-of-N (BoN) approach takes this process further by introducing an external verifier, typically a Reward Model (RM), to score, rank and select the best response from candidates (Stiennon et al., 2020; Irvine et al., 2023; Song et al., 2025). However, these prevailing strategies are inherently bounded by the quality of the set of candidates (Khairi et al., 2025). They are limited to produce a solution that transcends the quality of candidate proposals, which becomes particularly problematic when all candidates are flawed (Zhang et al., 2025).

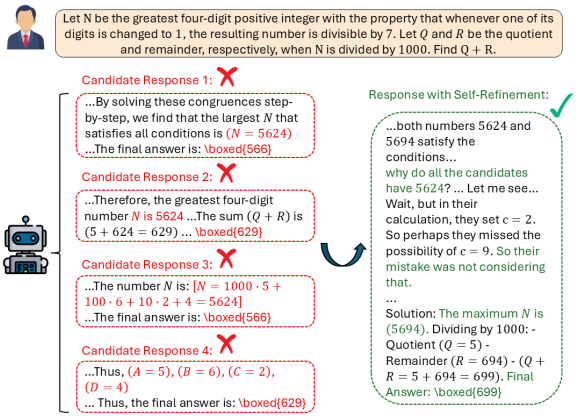


Figure 1: Case study of Qwen2.5-7B-Instruct after training. Even provided with four incorrect candidates, it can still reference them, diagnose the flaws, and finally construct a correct answer. We provide a more detailed case in Appendix F.

To address this limitation, several refinement methods (Madaan et al., 2023; Kim et al., 2023; Paul et al., 2024), where a model critiques and improves its own output, provide a promising alternative. These works have often focused on sequential refinement, where the model iteratively refines its answers. However, Huang et al. (2024) finds that LLMs struggle to self-refine their own reasoning without external feedback and in some cases

*Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 25904–25921
July 2-7, 2026 ©2026 Association for Computational Linguistics

can even degrade performance, highlighting the limitations of intrinsic sequential self-refinement. Consequently, recent research has shifted towards parallel aggregation methods (Vernikos et al., 2024; Zhang et al., 2025; Li et al., 2025b), attempting to synthesize a correct solution from multiple generated candidate solutions. While these approaches leverage candidate diversity, the fundamental mechanism behind the success remains obscure. It is unclear whether the improvement in parallel refinement is driven by better candidate coverage, enhanced reasoning abilities, or emergent interaction effects among multiple samples. This gap in understanding motivates us to develop a metric for evaluating intrinsic refinement ability. Therefore, we propose **the Refinement Gap**, which quantifies the improvement of refinement beyond majority voting. Based on this metric, we reveal a surprising systematic scaling trend: **the refinement capability is primarily governed by model scale, with minimal correlation to domain-specific capabilities.**

To address this gap, we introduce **Generative Self-Refinement (GSR)**, a training-enabled parallel test-time scaling framework designed to explicitly activate self-refinement in smaller models. GSR distills a refinement policy from high performance teacher models with a large refinement gap into smaller student models, enabling them to generate improved solutions beyond simple selection. Crucially, we employ **a hybrid training pipeline** that jointly optimizes for direct solution generation and refinement. This hybrid training pipeline ensures a synergy where strong base reasoning improves the refinement process, while refinement feedback enhances reasoning robustness. As illustrated in Figure 1, we empirically demonstrate that GSR enables small models to transcend the limits of selection, successfully recovering correct solutions even in the difficult scenarios where all candidates are incorrect.

Our key contributions are as follows:

- We introduce the Refinement Gap, a metric that systematically quantifies the intrinsic gain of self-refinement beyond majority voting. This metric reveals a scaling trend in refinement ability across model sizes.
- We propose Generative Self-Refinement (GSR), a training-enabled parallel test-time scaling framework that transfers the refinement policy from larger teacher models to smaller models.
- We demonstrate that our method achieves state-of-the-art results empirically on five challenging mathematical benchmarks over other parallel aggregation methods and refinement skill is robust under challenging candidate conditions.

2 Scaling Trend of Refinement Gap

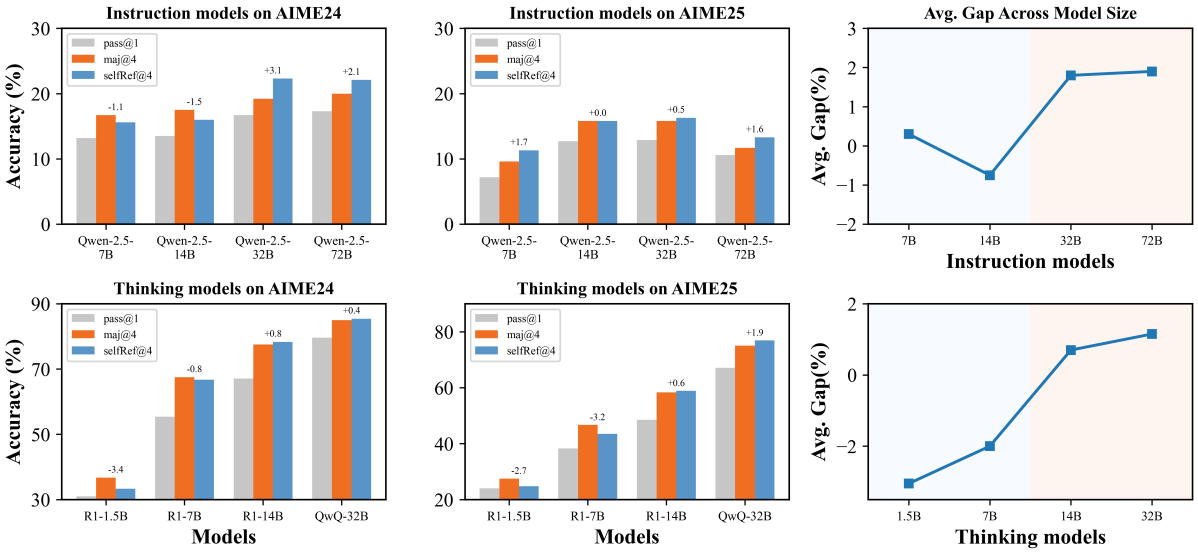
Recent works (Vernikos et al., 2024; Zhang et al., 2025; Li et al., 2025b) have considered parallel self-refinement as an effective method of test-time scaling. However, the underlying source of parallel self-refinement gain is still underexplored. In particular, even a simple self-consistency aggregation (majority voting) over the same candidate solutions can substantially improve accuracy, making it unclear how much improvement should be attributed to refinement procedure. This motivates our first research question: (**RQ1**) *Do we really need parallel self-refinement?*

To quantify the marginal gain of parallel self-refinement, we define the **Refinement Gap** as

$$\Delta(\mathcal{M}, \mathcal{D}, k) = \text{selfRef@k}(\mathcal{M}, \mathcal{D}) - \text{maj@k}(\mathcal{M}, \mathcal{D}) \quad (1)$$

for a given model $\mathcal{M}$, dataset $\mathcal{D}$ and candidate budget k. We use majority voting as *a natural gold baseline* because it is a deterministic function operator over the same solutions and is expected to improve monotonically with pass@1 under the independence assumptions (Kuncheva, 2014). We conduct a systematic study across model scales from 1.5B to 72B across multiple model families including instruction Qwen-2.5-Instruct (Qwen-2.5) and thinking DeepSeek-R1-Distill-Qwen/QwQ (R1/QwQ), on the competitive mathematics benchmarks AIME24/AIME25, to enable a comprehensive analysis of the effectiveness of parallel self-refinement.

As illustrated in Figure 2, we summarize the key observations as follows: **(1) Parallel self-refinement consistently raises the ceiling beyond the base capability measured by pass@1 across all models and benchmarks evaluated. (2) The Refinement Gap Δ exhibits a clear scaling trend across instruction and thinking models.** Further, we empirically observe *a regime shift* around 14B. Both instruction and thinking models $\geq$ 14B tend to achieve a higher average refinement gain $\overline{\Delta}$ than



(a) Performance on AIME24 (b) Performance on AIME25 (c) Avg. Gap Across Model Size

Figure 2: A comprehensive study of refinement gap. (***Left***) Accuracy of instruction models on AIME24/AIME25 under pass@1, majority voting (maj@4), and parallel self-refinement (selfRef@4). Numbers denote Δ = selfRef@4 − maj@4. (***Middle***) Accuracy of thinking models on AIME24/AIME25. (***Right***) Average $\overline{\Delta}$ across AIME24/AIME25 versus model size. Shaded regions highlight low and high gain according to $\overline{\Delta}$.

smaller models. We visualize these low and high gain regimes with different shaded regions in Figure 2. **(3) The Refinement Gap Δ is weakly correlated with the base capability of the models.** For instance, on AIME24, DeepSeek-R1-Distill-Qwen-7B attains a higher base capability (54.4%) yet has $\overline{\Delta} = -0.8$, while Qwen2.5-32B-Instruct attains a larger $\overline{\Delta} = 3.1$ despite a much lower pass@1(18.6%).

3 Methodology

3.1 Overview

Based on the analysis in Section 2, we identify that the refinement gap Δ exhibits a clear scaling trend. Empirically, smaller models often reside in a low gain regime, whereas larger models consistently exhibit a much higher refinement gap. Crucially, this capability cannot be reliably elicited in smaller models through prompting, motivating our hypothesis that **self-refinement is a learnable inference strategy that must be explicitly activated via training**.

To bridge this gap, we propose **Generative Self-Refinement (GSR)**, a parallel test-time scaling framework designed to transfer the self-refinement capability from larger models with higher gain to smaller models. At test time, GSR follows a generate-then-refine procedure, where a unified model samples k candidate solutions and then synthesizes a self-refined final answer.

3.2 Data Construction Pipeline

Guided by Section 2, we select large teacher models with high refinement gap to ensure reliable refinement supervision for small models. To this end, we construct refinement triples through the following steps as illustrated in Figure 3. For each problem x, we first sample k candidate solutions $\mathcal{C}_k = \{c_i\}_{i=1}^{k}$ from the student itself to match the test-time candidate distribution. We then build an augmented input $x_{aug} = \text{Aug}(x, \mathcal{C}_k)$ by concatenating the question with the enumerated candidates (see Appendix A and B for more details). Subsequently, we query the teacher to produce a refined target solution y_{ref}^{T}. This procedure yields the refinement dataset $\mathcal{D}_{\text{ref}} = \{(x^{(i)}, \mathcal{C}_k^{(i)}, y_{\text{ref}}^{T\,(i)})\}_{i=1}^{N}$. Unlike vanilla distillation, this supervision teaches how to refine the student's own candidates, rather than only imitating direct answers. Notably, to ensure the student learns synthesis beyond candidate responses selection, we intentionally retain hard instances where the candidate pool is of low quality and highly inconsistent. Specifically, we emphasize the cases with $N_c = 0$ (i.e., all candidate responses are incorrect) as well as other highly inconsistent scenarios, where voting or ranking is not effective. These critical cases provide direct supervision for constructing a correct solution by

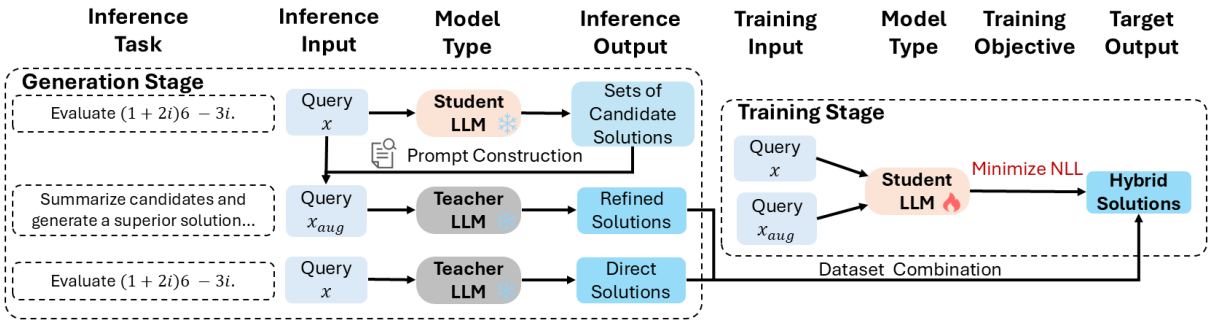


Figure 3: An overview of the hybrid training pipeline, which consists of a data generation stage (***Left***) followed by a supervised fine-tuning (SFT) stage (***Right***). We use a teacher model to construct hybrid dataset and then train the student model on dual tasks.

integrating partial insights and correcting errors from imperfect candidates.

In addition to the refinement dataset $\mathcal{D}_{\text{ref}}$, we collect direct solutions to maintain and strengthen the base capability of student. Concretely, we query the teacher with the original question x to obtain a direct target solution y^T_{dir} and forms a direct-solving dataset $\mathcal{D}_{\text{dir}} = \{(x^{(i)}, y^{T\ (i)}_{\text{dir}})\}_{i=1}^{N}$. This component is crucial because the candidate quality sets the performance ceiling for refinement and stabilizes the joint optimization in Section 3.3.

3.3 Hybrid Training Objective

We optimize a single student model $\mathcal{S}_\theta$ to acquire two complementary skills, **direct-solving** for generating strong candidates and **refinement** for synthesizing a better final solution given these candidates, aiming for a higher performance ceiling. As shown in Figure 3, we optimize model $\mathcal{S}_\theta$ on the direct-solving dataset $\mathcal{D}_{\text{dir}}$ and the refinement dataset $\mathcal{D}_{\text{ref}}$. Formally, we minimize the negative log-likelihood (NLL) objectives as follows:

$$\begin{aligned}\mathcal{L}_{\text{dir}}(\theta) &= -\mathbb{E}_{(x,y^T_{\text{dir}})\sim\mathcal{D}_{\text{dir}}}[\log \mathcal{S}_\theta(y^T_{\text{dir}}|x)] \\ \mathcal{L}_{\text{ref}}(\theta) &= -\mathbb{E}_{(x,\mathcal{C}_k,y^T_{\text{ref}})\sim\mathcal{D}_{\text{ref}}}[\log \mathcal{S}_\theta(y^T_{\text{ref}}|x_{\text{aug}})]\end{aligned} \tag{2}$$

Then, we optimize a weighted combination:

$$\mathcal{L}(\theta) = \lambda\mathcal{L}_{\text{ref}}(\theta) + (1-\lambda)\mathcal{L}_{\text{dir}}(\theta) \tag{3}$$

where λ controls the trade-off between learning refinement behavior and improving base capability. For simplicity, we set $\lambda = 0.5$.

Joint optimization for hybrid objectives is essential to the success of GSR. Training only the refinement objective tends to be bottlenecked by candidate quality, which in turn limits the headroom. In contrast, training only the direct-solving objective improves the base capability but does not reliably teach the model to leverage imperfect candidates, resulting in a low or unstable refinement gap Δ. By jointly optimizing direct solving and refinement, the model preserves strong base capability while learning the refinement strategy, getting complementary gains in both candidate quality and refinement effectiveness. At test time, the unified model $\mathcal{S}_\theta$ performs generate-then-refine by sampling k candidates and synthesizing a refined answer **without any external module**.

4 Experiments

4.1 Experimental Setup

Dataset Curation We construct our training corpus from OpenMathReasoning (Moshkov et al., 2025), a large math dataset generated by Deepseek-R1 (DeepSeek-AI, 2025) and QwQ-32B(QwenTeam, 2025). We retain 184k direct-solving instances and derive an additional 184k refinement instances, where each instance pairs the original query with four candidate responses. Details are provided in Appendix E.

Training Settings We perform supervised fine-tuning (SFT) on Qwen2.5-7B-Instruct (Yang et al., 2025b) and denote the resulting model as **GSR-7B**. We present more training details in Appendix D.1.

Aggregation Baselines To rigorously evaluate the efficacy, we compare with the following aggregation TTS baselines: (1) **Majority Voting** (Wang et al., 2023), a method selects the most consistent answer without any external modules. (2) **Skywork-Reward-Gemma-2-27B-v0.2** (SkyworkRM-27B) (Liu et al., 2024), a SOTA scalar reward model which ranks the overall quality without explanations or reasoning. (3) **RM-**

Table 1: Comprehensive performance evaluation on five mathematical benchmarks. We report the results and the Refinement Gap Δ (selfRef - maj) of our GSR-7B and base Qwen2.5-7B-Instruct for reference. Additionally, we report the pass@1 metric for two leading models from Yang et al. (2025c).

Method	Aggregation Model	AIME24	AIME25	AMC22-23	MATH500	Olympiad	Avg.
Larger Models							
QwQ-Preview	-	50.0			90.6	61.2	
o1-mini	-	56.7			90.0		
Qwen2.5-7B-Instruct (*Base*)							
pass@1	-	13.2	7.2	43.6	75.9	39.5	35.9
maj@4	-	16.7	9.6	47.7	79.4	43.5	39.4
BoN@4	SkyworkRM-27B	17.1	9.2	48.0	78.6	43.0	39.2
selfRef@4	-	15.6	11.3	47.5	78.7	43.8	39.4
Δ		-1.1	+1.7	-0.2	-0.7	+ 0.3	**±0.0**
GSR-7B (*After Training*)							
pass@1	-	50.1	37.8	78.5	90.6	64.4	64.3
maj@4	-	60.0	46.7	84.6	92.8	68.3	70.5
maj@5	-	60.7	48.3	84.9	93.1	68.8	71.1
BoN@4	RRM-7B	60.4	46.3	80.4	90.6	64.6	68.5
BoN@4	RM-R1-7B	62.1	45.4	84.6	**93.5**	69.1	70.9
BoN@4	SkyworkRM-27B	58.3	46.7	81.6	92.5	67.6	69.3
BoN@5	SkyworkRM-27B	59.2	46.3	82.8	92.7	67.7	69.7
Ref@4	Synthesizer-8B	60.2	44.0	84.0	92.0	67.7	69.6
selfRef@4	-	**66.0**	**51.7**	**85.7**	93.4	**71.0**	**73.6**
Δ		+ 6.0	+ 5.0	+ 1.1	+ 0.6	+ 2.7	**+3.1**

R1-DeepSeek-Distilled-Qwen (Chen et al., 2025) (RM-R1) and **RRM** (Guo et al., 2025), two concurrent approaches integrating reasoning capabilities into reward modeling, significantly surpassing conventional RM's performance. (4) **Synthesizer-8B-math** (Synthesizer-8B) (Zhang et al., 2025), an aggregation method which synthesizes multiple candidates to produce a final answer, achieving SOTA performance on math domain.

Benchmarks For a comprehensive evaluation of mathematical performance, we evaluate all baselines across five challenging and most representative benchmarks (Hochlehnert et al., 2025) in the mathematical domain: AIME24 (AI-MO, 2024a), AIME25 (Lin, 2025), AMC22 & AMC23 (AI-MO, 2024b), MATH500 (Hendrycks et al., 2021) and OlympiadBench (He et al., 2024).

Evaluation Settings For a direct and fair comparison, we report the average metrics maj@4 (majority voting), BoN@4 (Best-of-N) and Ref/SelfRef@4 (Refinement/Self-Refinement) on the same sets of four candidate responses ($k = 4$). To ensure a fair comparison of computational budget, we additionally report the average metrics for majority voting and non-generative BoN methods with five candidates ($k = 5$). We report the average metrics of 32 runs for AIME24 and AIME25 and 16 runs for the remaining. More details can be found in Appendix D.2.

4.2 Main Results

We present a comprehensive evaluation of our method against advanced parallel test-time scaling baselines. The results, summarized across five challenging mathematical benchmarks, are presented in Table 1. A primary observation is the substantial performance shift after the training process. Compared to the base model, GSR-7B exhibits dramatic improvements across all standard metrics, particularly on the most challenging benchmarks. For instance, its pass@1 accuracy on AIME24 increases from a modest 13.2% to 50.1%, confirming the efficacy of our approach.

Our method shows significant improvement after post-training. Before training, the performance of selfRef@4 on the base model is comparable with majority voting. After training, our method emerges as the **state-of-the-art (SOTA)** method, achieving 73.6% average accuracy. It surpasses not only standard baselines such as majority voting but also more complex Best-of-N (BoN@4), as demonstrated on AIME24 (66.0% vs. 62.1%). **GSR-7B even achieves a higher average Refinement Gap Δ on AIME24/AIME25 (+5.5%) than strong teacher model QwQ-32B (+1.15%).** We

note that in the MATH500 dataset, selfRef@4 is marginally underperformed by BoN@4 (93.4% vs. 93.5%). We hypothesize that it is due to the specific dynamics of high-accuracy regimes (90.6% on pass@1). In such a scenario, at least a correct answer is highly likely to be present in any set of candidate responses, making selection methods like Best-of-N particularly effective.

4.3 Ablation Study on Training

In this section, we conduct an ablation study to answer: (**RQ2**) *Does our hybrid training pipeline improve the upper bound of accuracy more than single mode training pipeline (direct-solving or refinement)?* To ensure a fair comparison, we construct three distinct training datasets randomly sampled from the total dataset, all of which contain 20k samples and share an identical set of questions: (1) Direct-Solving (DS): 20k samples from $\mathcal{D}_{\text{dir}}$, also denoted as a vanilla knowledge distillation baseline. (2) Refinement (Ref): 20k samples from $\mathcal{D}_{\text{ref}}$. (3) Ours: a balanced mix of 10k from $\mathcal{D}_{\text{dir}}$ and 10k from $\mathcal{D}_{\text{ref}}$.

Table 2: Ablation study on different training strategies.

Method	AIME24		AIME25	
	pass@1	selfRef@4	pass@1	selfRef@4
baseline	13.2	15.6	7.2	9.6
+*DS*	**38.3**	44.3	**29.6**	33.2
+*Ref*	26.4	45.0	23.5	30.0
+*ours*	37.5	**51.5**	29.0	**36.5**

The results presented in Table 2 lead to a clear conclusion. The model trained on a DS dataset achieves the highest pass@1 scores while **its' capacity for self-refinement is severely limited.** To our surprise, even with training purely on the Ref dataset, the model still **achieves a higher pass@1 score than the base model.** However, this model excels at the self-refinement task but demonstrates **subpar pass@1 performance compared with two others.** Our method strikes the optimal balance. It achieves the highest selfRef@4 scores across both benchmarks while maintaining a highly competitive pass@1 performance, confirming the necessity of the hybrid training pipeline.

4.4 Fine-Grained Analysis on Self-Refinement

Although the overall accuracy metrics are informative in Section 4.2, the underlying mechanism of our approach is still obscure. We conduct a more fine-grained conditional analysis across all methods when provided with imperfect or even entirely incorrect candidates. We report the average accurate rates conditioned on the number of correct candidates out of 4 ($N_c \in \{0, 1, 2, 3, 4\}$).

Comparison with Baselines As shown in Figure 4, when a clear majority of candidates are correct ($N_c \geq 3$), all methods perform exceptionally well, often achieving near perfect accuracy. However, our methods begin to differentiate in more ambiguous cases ($N_c < 3$). When only one candidate is correct, our selfRef@4 achieves an impressive 60.2% accuracy, substantially outperforming majority voting (maj@4 at 18.8%) and all BoN@4 variants. Crucially, even when all candidates are incorrect ($N_c = 0$), **our approach can still produce a correct solution 5.9% of the time, whereas all baseline methods fail.** These low N_c scenarios serve as a proxy for a class of most challenging problems where generating a correct solution is difficult. The effectiveness of our method in these situations highlights its robustness for solving difficult problems.

Generalization across Benchmarks As shown in Figure 5, GSR-7B is able to recover a correct solution from a complete set of incorrect candidates ($N_c = 0$) across all benchmarks, ranging from 3.4% on MATH500 to 9.0% on AMC22-23. This consistency proves the refinement capability is a generalizable skill that GSR-7B has acquired.

4.5 Robustness and Generalization Analysis

In this section, we conduct analytic experiments on two larger models, Qwen2.5-14B/32B-Instruct (Yang et al., 2025b) to address two critical questions: (**RQ3**) *Is our method effective across different model scales?* (**RQ4**) *Does the model learn a generalizable improvement skill, or does it merely learn to correct its own specific errors?* We use the same hybrid training dataset and experimental setup as described in Section 4.3.

Robustness Across Different Model Sizes The results, presented in Table 3, demonstrate the effective scalability of our approach to models of a larger-scale. After training, both the 14B and 32B models exhibit substantial performance improvements across standard metrics (pass@1 and maj@4) compared to their w/o SFT counterparts. More critically, the results demonstrate the effectiveness of our method. For fine-tuned models,

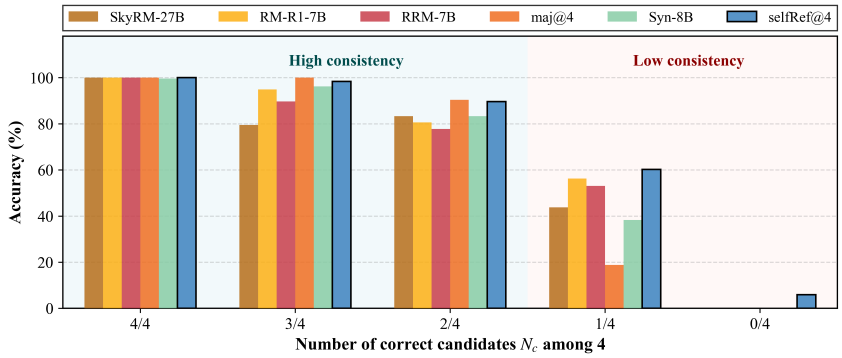


Figure 4: Fine-grained analysis on the AIME24. We report accurate rates in percentage (%) conditioned on the number of correct candidates (N_c).

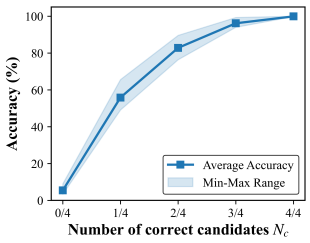


Figure 5: Robustness analysis of our method across five diverse benchmarks. The full details are provided in Appendix C Table 5.

Table 3: Performance on AIME benchmarks across different model scales. The table compares two fine-tuned models (w training) against their base counterparts (w/o training).

Method	Qwen2.5-14B-Instruct		Qwen2.5-32B-Instruct	
	AIME24	AIME25	AIME24	AIME25
w/o Training				
pass@1	13.5	12.7	16.7	12.9
maj@4	**17.5**	**15.8**	19.2	15.8
selfRef@4	16.0	**15.8**	**22.3**	**16.3**
w Training				
pass@1	49.4	36.9	66.3	54.2
maj@4	56.7	41.7	72.5	59.2
selfRef@4	**68.1**	**49.6**	**75.2**	**67.3**

selfRef@4 consistently outperforms their respective majority voting baselines. For instance, the fine-tuned 14B model achieves a 20.1% relative improvement (68.1% vs. 56.7%) in AIME24, while the 32B model shows a 13.7% relative improvement (67.3% vs. 59.2%) in AIME25. This phenomenon contrasts with the 14B base model, which fails to benefit from self-refinement and even shows a performance degradation in AIME24 compared to the majority voting baseline.

Generalization via Data Decoupling A crucial aspect of this experiment is that we reuse the hybrid dataset from Section 4.3 to train larger models. In particular, **all candidate solutions within this dataset are generated by Qwen2.5-7B-Instruct**. Although there is a mismatch between the candidate generator and learners, the results clearly indicate that both models successfully acquire the ability of self-refinement. This finding provides compelling evidence for a decoupling of the model's internal knowledge representation from the self-refinement skill.

4.6 Input Scaling Analysis

To scale up to a larger number of inputs, previous work (Zhang et al., 2025; Guo et al., 2025) has often employed a hierarchical strategy, partitioning candidate responses into fixed-size groups, producing outputs for each group, and then iteratively combining across groups. This approach has been shown to be general and effective. However, such methods are not indeed scaling on input and have a time complexity of $\mathcal{O}(N)$.

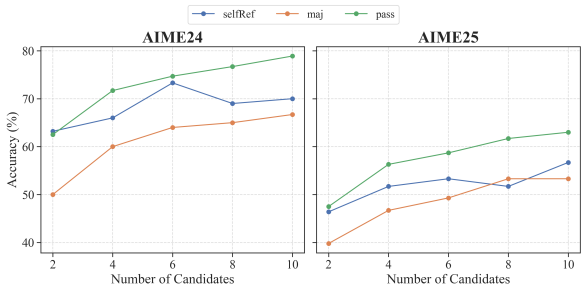


Figure 6: Accuracy of input scaling on the performance across the (***Left***) AIME24 and (***Right***) AIME25 benchmarks.

Our experiments are conducted on **a fixed set of four candidate responses**. To evaluate our model's extrapolation capability, we conduct an input scaling experiment. Specifically, we evaluate performance with the number of candidate responses, k, ranging from 2 to 10. We compare selfRef@k (self-refinement) with the maj@k (majority voting) and oracle pass@k. In Figure 6, for a smaller number of candidates ($k \leq 4$), GSR-7B shows strong performance, even slightly **surpassing the pass@2** (63.2% vs. 62.5%) on AIME24 at $k = 2$. For a larger number of candidates, our method consis-

tently outperforms majority voting, and its performance scales with the number of candidates. This suggests that our method effectively extrapolates to a number of candidates greater than four.

4.7 Out-of-Distribution Analysis

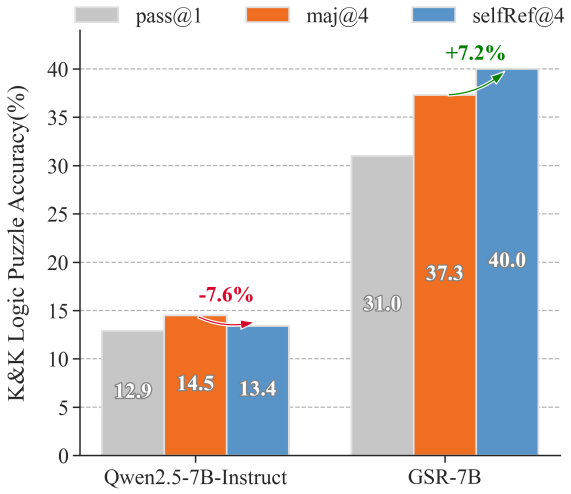


Figure 7: Experiments results for our model and its base model on a subset of K&K dataset. The arrows and percentages quantify the relative performance change when applying our method over majority voting.

GSR-7B is trained exclusively on math datasets. To further analyze its generalization ability to an unseen domain, we test it on the Knights and Knaves (K&K) logic puzzles dataset (Xie et al., 2025), where some characters tell the truth and others only lie. For evaluation, we use the 4ppl subset of the K&K dataset and reformat original problems as multiple-choice questions to allow more accurate scoring. We report the performance of pass@1, majority voting (maj@4) and our method (selfRef@4). In Figure 7, GSR-7B demonstrates a substantial performance improvement over the base model. Compared to the base model's pass@1 accuracy of 12.9%, our model achieves 37.3%. More critically, we observe a clear divergence in self-refinement ability. The base model's selfRef@4 performance decreases by 7.6% relative to maj@4, while our model shows a performance improvement of 7.2%. These results confirm that the self-refinement ability is a skill acquired during training and that this skill even successfully generalizes to an unseen out-of-distribution domain.

5 Related Work

Aggregation Methods The most well-known approach within parallel aggregation is majority voting (Wang et al., 2023), which leverages the self-consistency of the model's outputs to select the most consistent answer. Best-of-N (BoN) replaces the voting mechanism with an external verifier (Reward Model, RM), which is used to either assign a scalar score (Cobbe et al., 2021; Rafailov et al., 2024; Winata et al., 2025), or to perform discriminative methods to indicate a preference (Stiennon et al., 2020; Nakano et al., 2022). Some works have introduced the Generative Reward Model (GenRM), which leverages the model's generative capacity to provide a detailed and interpretable justification (Kim et al., 2024; Mahan et al., 2024; Whitehouse et al., 2025; Chen et al., 2025; Guo et al., 2025).

Refinement Methods Some studies have explored that models can refine responses to improve the performance (Cobbe et al., 2021; Gou et al., 2024; Ferraz et al., 2024). Self-Refine (Madaan et al., 2023), RIC (Kim et al., 2023) and REFINER (Paul et al., 2024) focus on sequential self-refinement, where a single model is used to generate feedback on its own output and iteratively revise the solution accordingly. Distinct from sequential approaches, LLM-Blender (Jiang et al., 2023), LM-Cor (Vernikos et al., 2024), AFT (Li et al., 2025a), CoT-based-Synthesizer (Zhang et al., 2025) train a dedicated generative model to fuse multiple solutions. However, these methods focus exclusively on the task of improving parallel solutions, while neglecting to train the model to solve problems directly. Besides, Zhao et al. (2025) learns solution aggregation as an explicit skill via reinforcement learning, showing that majority voting is not always optimal. MoA (Wang et al., 2025) and Multiagent FT (Subramaniam et al., 2025) leverages multi-agent society of specialized LLMs to improve responses, incurring significant inference and deployment overhead.

6 Conclusion

In this paper, we systematically investigate the scaling properties of the gains of self-refinement over majority voting in LLMs. By introducing Refinement Gap, we provide a clear scaling trend: relative refinement gap increases with model size and is only weakly related to base capability. Motivated by this, we propose Generative Self-Refinement (GSR), a parallel TTS framework that transfers refinement ability from teacher models to smaller student models. GSR jointly optimizes for direct

solving and refinement objectives, enabling a single and unified model to generate strong candidates and synthesize a better final answer even from imperfect candidates. Empirically, GSR substantially improves small models reasoning performance and demonstrates robust refinement behavior under challenging candidate conditions, highlighting refinement as a learnable inference strategy that can be explicitly activated via training.

7 Limitations

Due to limited computational resources, our experiments focus on mathematical reasoning domains. Although this setting is natural for studying self-refinement and test-time scaling, it may not fully represent other reasoning scenarios (e.g., logical deduction, programming, scientific QA, or agentic tool use). We plan to conduct more experiments across a broader range of reasoning domains in the future work.

Our data construction pipeline relies on strong teacher models and the verifiable reward to filter solutions. This design may be less directly applicable to tasks that lacks reliable verification. Extending our approach to such settings likely requires alternative supervision signals, such as human annotation, preference-based feedback, or external verifiers/tools, which may introduce additional cost and noise.

References

AI-MO. 2024a. AIMO Validation AIME Dataset. https://huggingface.co/datasets/AI-MO/aimo-validation-aime. Accessed: 2025-03-29.

AI-MO. 2024b. AIMO Validation AMC Dataset. https://huggingface.co/datasets/AI-MO/aimo-validation-amc. Accessed: 2025-03-29.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *Preprint*, arXiv:2407.21787.

Xiusi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru Wang, Yu Zhang, Denghui Zhang, Tong Zhang, Hanghang Tong, and Heng Ji. 2025. Rm-r1: Reward modeling as reasoning. *Preprint*, arXiv:2505.02387.

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. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Thomas Palmeira Ferraz, Kartik Mehta, Yu-Hsiang Lin, Haw-Shiuan Chang, Shereen Oraby, Sijia Liu, Vivek Subramanian, Tagyoung Chung, Mohit Bansal, and Nanyun Peng. 2024. LLM self-correction with decrim: Decompose, critique, and refine for enhanced following of instructions with multiple constraints. In *Findings of Proceedings of EMNLP 2024*, pages 7773–7812. Association for Computational Linguistics.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. CRITIC: large language models can self-correct with tool-interactive critiquing. In *Proceedings of ICLR 2024*.

Jiaxin Guo, Zewen Chi, Li Dong, Qingxiu Dong, Xun Wu, Shaohan Huang, and Furu Wei. 2025. Reward reasoning model. *Preprint*, arXiv:2505.14674.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. Olympiadbench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of ACL 2024*, pages 3828–3850. Association for Computational Linguistics.

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. In *Proceedings of NeurIPS 2021*.

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.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. In *Proceedings of ICLR 2024*.

Robert Irvine, Douglas Boubert, Vyas Raina, Adian Liusie, Ziyi Zhu, Vineet Mudupalli, Aliaksei Korshuk, Zongyi Liu, Fritz Cremer, Valentin Assassi, Christie-Carol Beauchamp, Xiaoding Lu, Thomas Rialan, and William Beauchamp. 2023. Rewarding chatbots for real-world engagement with millions of users. *Preprint*, arXiv:2303.06135.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of ACL 2023*, pages 14165–14178. Association for Computational Linguistics.

Ammar Khairi, Daniel D'souza, Marzieh Fadaee, and Julia Kreutzer. 2025. Making, not taking, the best of n. *Preprint*, arXiv:2510.00931.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2023. Language models can solve computer tasks. In *Proceedings of NeurIPS 2023*.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2024. Prometheus 2: An open source language model specialized in evaluating other language models. In *Proceedings of EMNLP 2024*, pages 4334–4353. Association for Computational Linguistics.

Ludmila I Kuncheva. 2014. *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons.

Yafu Li, Zhilin Wang, Tingchen Fu, Ganqu Cui, Sen Yang, and Yu Cheng. 2025a. From drafts to answers: Unlocking llm potential via aggregation fine-tuning. *Preprint*, arXiv:2501.11877.

Zichong Li, Xinyu Feng, Yuheng Cai, Zixuan Zhang, Tianyi Liu, Chen Liang, Weizhu Chen, Haoyu Wang, and Tuo Zhao. 2025b. Llms can generate a better answer by aggregating their own responses. *Preprint*, arXiv:2503.04104.

Yen-Ting Lin. 2025. Aime 2025 dataset. `https://huggingface.co/datasets/yentinglin/aime_2025`. Accessed: 2025-03-29.

Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. 2024. Skywork-reward: Bag of tricks for reward modeling in llms. *Preprint*, arXiv:2410.18451.

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. In *Proceedings of NeurIPS 2023*.

Dakota Mahan, Duy Van Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato, Jan-Philipp Fränken, Chelsea Finn, and Alon Albalak. 2024. Generative reward models. *Preprint*, arXiv:2410.12832.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. 2025. Aimo-2 winning solution: Building state-of-the-art mathematical reasoning models with openmathreasoning dataset. *Preprint*, arXiv:2504.16891.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2022. Webgpt: Browser-assisted question-answering with human feedback. *Preprint*, arXiv:2112.09332.

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2024. REFINER: reasoning feedback on intermediate representations. In *Proceedings of EACL 2024*, pages 1100–1126. Association for Computational Linguistics.

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

Rafael Rafailov, Yaswanth Chittepu, Ryan Park, Harshit Sikchi, Joey Hejna, W. Bradley Knox, Chelsea Finn, and Scott Niekum. 2024. Scaling laws for reward model overoptimization in direct alignment algorithms. In *Proceedings of NeurIPS 2024*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *Preprint*, arXiv:2408.03314.

Yifan Song, Guoyin Wang, Sujian Li, and Bill Yuchen Lin. 2025. The good, the bad, and the greedy: Evaluation of llms should not ignore non-determinism. In *Proceedings of NAACL 2025*, pages 4195–4206. Association for Computational Linguistics.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F. Christiano. 2020. Learning to summarize with human feedback. In *Proceedings of NeurIPS 2020*.

Vighnesh Subramaniam, Yilun Du, Joshua B. Tenenbaum, Antonio Torralba, Shuang Li, and Igor Mordatch. 2025. Multiagent finetuning: Self improvement with diverse reasoning chains. In *Proceedings of ICLR 2025*.

Giorgos Vernikos, Arthur Brazinskas, Jakub Adámek, Jonathan Mallinson, Aliaksei Severyn, and Eric Malmi. 2024. Small language models improve giants by rewriting their outputs. In *Proceedings of EACL 2024*, pages 2703–2718. Association for Computational Linguistics.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. 2025. Mixture-of-agents enhances large language model capabilities. In *Proceedings of ICLR 2025*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *Proceedings of ICLR 2023*.

Chenxi Whitehouse, Tianlu Wang, Ping Yu, Xian Li, Jason Weston, Ilia Kulikov, and Swarnadeep Saha. 2025. J1: Incentivizing thinking in llm-as-a-judge via reinforcement learning. *Preprint*, arXiv:2505.10320.

Genta Indra Winata, David Anugraha, Lucky Susanto, Garry Kuwanto, and Derry Tanti Wijaya. 2025. Metametrics: Calibrating metrics for generation tasks using human preferences. In *Proceedings of ICLR 2025*.

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. 2025. On memorization of large language models in logical reasoning. *Preprint*, arXiv:2410.23123.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

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, Junyang Lin, Kai Dang, and 23 others. 2025b. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Ling Yang, Zhaochen Yu, Bin Cui, and Mengdi Wang. 2025c. Reasonflux: Hierarchical llm reasoning via scaling thought templates. *Preprint*, arXiv:2502.06772.

Bohan Zhang, Xiaokang Zhang, Jing Zhang, Jifan Yu, Sijia Luo, and Jie Tang. 2025. Cot-based synthesizer: Enhancing llm performance through answer synthesis. *Preprint*, arXiv:2501.01668.

Wenting Zhao, Pranjal Aggarwal, Swarnadeep Saha, Asli Celikyilmaz, Jason Weston, and Ilia Kulikov. 2025. The majority is not always right: Rl training for solution aggregation. *Preprint*, arXiv:2509.06870.

A Prompt Template

GSR supports dual abilities during inference: direct-answer generation and self-refinement. The choice between modes is achieved by constructing different input prompts. Here we provide the detailed prompt template used to prompt the model to perform self-refinement. The prompt template consists of the original problem and a set of candidate solutions mainly.

Prompt Template on the Refinement Mode

You are an expert and creative solver, provided with a challenging problem and a set of candidate responses which may be correct, partially correct or even wrong.
You should first fully summarize the connection between candidate responses and problem, then generate a new and superior solution. You should generate a correct solution yourself if all candidates are wrong. Don't copy candidates, use insights selectively and reason independently.
Problem:
{Problem}
Think step by step and put final answer within `\boxed{}`.

Candidate Response 1:
{Response 1}

Candidate Response 2:
{Response 2}

Candidate Response 3:
{Response 3}

Candidate Response 4:
{Response 4}

We also provide a common prompt template to instruct the model to solve the problem directly. This common prompt template is applicable for any LLMs.

Common Prompt Template for Direct-Answer Generation

{Problem}
Think step by step and put final answer within `\boxed{}`.

B Analysis of Candidate Input Burden

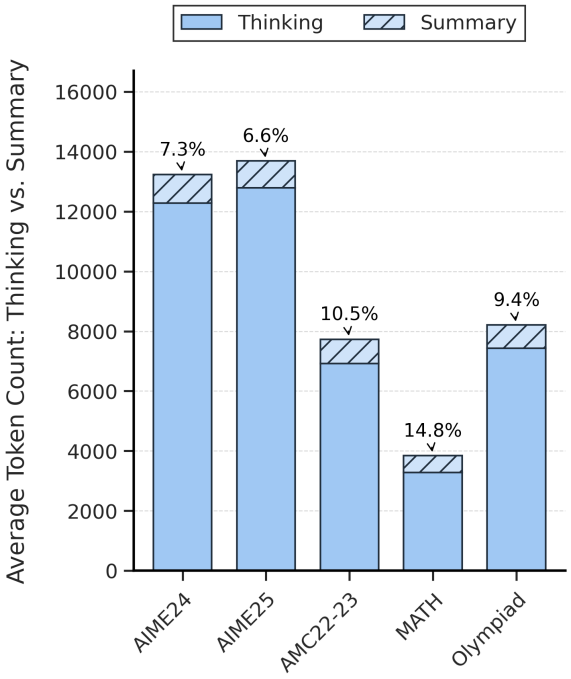


Figure 8: Average token counts of model **Direct-Solving** outputs across five benchmarks, showing the breakdown between the Thinking (solid) and Summary (hatched) components. Percentages indicate the proportion of summary tokens.

A critical factor for the feasibility of our method is the management of context length, as recontextualizing the verbose responses from thinking model can lead to prohibitive compute overhead. We innovatively mitigate this by exclusively **extracting the summary component from responses while discarding the preceding thinking component**.

To quantify the efficiency of this strategy, we analyze the token composition of our model's outputs across the five benchmarks of Section 4.2. As illustrated in Figure 8, the summary component is remarkably succinct, consistently accounting for only a small fraction of the total generated tokens (e.g., 6.6% to 14.8%). In particular, for difficult benchmarks like AIME24 and AIME25, the total number of tokens generated is substantial and exceeds 13,000 on average. Despite this, the summary component itself still remains remarkably concise, with an average token count of fewer than 1,000. Across the five distinct and representative benchmarks, the token length of the summary component varies from 570 to 960 tokens. This demonstrates that the input context is well controlled, imposing an average input burden of less than 4000 tokens even

Table 4: Average token counts of model **Self-Refinement** outputs against **Direct Solving** across five benchmarks.

Token Count \ Benchmark	AIME24	AIME25	AMC22-23	MATH500	Olympiad
self-Refinement	8878	10477	5270	2745	5910
Direct Solving	13232	13687	7732	3850	8209
Reduction(%)	-32.9%	-23.5%	-31.8%	-28.7%	-28.0%

on the challenging AIME24 benchmark.

As detailed in Table 4, we further calculate the average token length for GSR-7B to perform self-refinement given four candidate responses. We find that token consumption is significantly lower than the direct-solving approach. We attribute this efficiency to the model's ability to leverage valuable information from correct candidate responses and thus to prune the search space significantly during its chain-of-thought reasoning.

C Full Results of Fine-Grained Analysis

In this section, we provide the full experimental results in Section 4.4. The results of fine-grained analysis of our method across AIME24, AIME25, AMC22-23, MATH500, Olympiad is provided in Table 5. For each number of correct candidate responses (from 0 to 4), we compile statistics on our model's performance, including the number of correct answers, wrong answers, total trials, and the final correct ratio.

D Experiments Details

D.1 Training Settings

We train Qwen2.5-7B-Instruct(Yang et al., 2025b) on the 368K SFT dataset for 3 epochs with the AdamW optimizer, employing a 10% linear warmup followed by a cosine learning rate decay schedule. The maximum learning rate is set to 1e-4, with a batch size of 128 and a maximum sequence length of 24K tokens. To support longer context windows and align with advanced thinking mode, we adopted the chat_template from the Qwen3 family(Yang et al., 2025a) and extended the maximum sequence length by setting max_position_embeddings to 65,536.

In the ablation study, we train the model for 5 epochs and set the maximum learning rate to 4e-5. All other settings remain the same.

D.2 Evaluation Settings

We explicitly instruct all models to think step by step and to enclose the final answer within `\boxed{}` (Hochlehnert et al., 2025). We use the Math-Verify framework, a more robust way to extract and verify answers.

We first sample candidate responses in the direct-solving mode and then perform various test-time scaling strategies based on these candidate responses. For base Qwen2.5-7B-Instruct, we configure the evaluation process to set the maximum new tokens of 4,096 and apply optimal hyperparameters. For our GSR-7B, we configure the maximum new tokens of 32,768, temperature of 0.6 and top-p 0.95. To mitigate the evaluation variance (Hochlehnert et al., 2025), we repeat 32 trials for every problem in AIME24 and AIME25, and 16 for all other benchmarks. We present the average accuracy across 16 or 32 samples generated directly for every problem as pass@1.

Then, we compute several TTS metrics based on non-overlapping groups of every four candidate responses: maj@4 (majority voting accuracy), BoN@4 (Best-of-N) and Ref/SelfRef@4. The final scores for these metrics are the average performance across all groups. For all baseline models, we follow the exact hyperparameter values and the specific prompt recommended in their official documentation or model cards. For RRM and RM-R1, we employ a knockout tournament strategy (Guo et al., 2025), a method of iterative pairwise comparison and elimination to determine the best answer, which effectively guides LLMs to perform BoN sampling. Our method is evaluated with a maximum of 32,768 output tokens, 6,144 tokens for candidate response input (1,566 tokens per candidate response), temperature of 0.6 and top-p 0.95. We ensure that our method also runs 32 trials for AIME24 and AIME25, and 16 for the remaining benchmarks.

Table 5: Full results of robustness analysis of our method across five mathematical benchmarks. We report correct number, wrong number, correct ratio and total conditioned on the number of correct candidates.

Number of Correct Candidates	Statistics	Benchmarks				
		AIME24	AIME25	AMC22-23	MATH500	Olympiad
4/4	Correct	260	200	867	6604	5573
	Wrong	0	0	1	12	11
	Total	260	200	868	6616	5584
	Correct Ratio(%)	100	100	99.9	99.8	99.8
3/4	Correct	152	92	147	581	1049
	Wrong	4	4	1	35	67
	Total	156	96	148	616	1116
	Correct Ratio(%)	97.4	95.8	99.3	94.3	94.0
2/4	Correct	129	101	80	165	564
	Wrong	15	15	20	51	132
	Total	144	116	100	216	696
	Correct Ratio(%)	89.6	87.1	80	76.4	81.0
1/4	Correct	77	84	30	112	367
	Wrong	51	44	26	116	357
	Total	128	128	56	228	724
	Correct Ratio(%)	60.2	65.6	53.6	49.1	50.7
0/4	Correct	16	19	14	11	117
	Wrong	256	401	142	313	2563
	Total	272	420	156	324	2680
	Correct Ratio(%)	5.9	4.5	9.0	3.4	4.4

E Dataset Curation

For simplicity, we establish a data pipeline based on a single large-scale mathematics data set, OpenMathReasoning(Moshkov et al., 2025). OpenMathReasoning dataset is curated from the AoPS community forums, performed rigorous filtering, classification, transformation, and benchmark decontamination, containing 540K mathematical problems and 3.2M generations from DeepSeek-R1 and QwQ-32B.

The detailed procedure, including the numbers of problems and samples remaining at each stage, is documented in Table 6. In summary, the pipeline involves several filtering steps, such as removing generations not used in the Kaggle competition, removing problems without an extracted answer, and discarding any generations without a pass rate evaluated by Qwen2.5-72B-Instruct. Subsequently, we aggregate the corresponding solutions for each problem, and then retaining problems with a pass rate between 0.25 and 0.9, yielding a prefiltered pool of 56K unique problems and 617K generations.

We construct the self-refinement dataset ($\mathcal{D}_{selfR}$) using these 56K problems. As summarized in Table 7, for each problem, we generate 6 candidate responses with Qwen2.5-7B-Instruct using temperature of 1.0, top-p of 0.95, and a maximum output length of 4096 tokens to foster diversity. Given the inherent difficulty of the problems, we implement a specific selection process for the 6 responses. We filter and construct a fixed size set of 4 candidate responses. This set is composed of all correct solutions from the initial candidate pool of 6, with the remaining filled by incorrect solutions to meet the required size of 4. The order of these 4 candidates is then randomized. By combining each original problem with its set of candidate responses according to the prompt template in Appendix A, we constructed a dataset of 56K self-refinement problems. We further remove data where the prompt length exceeded 8,192 tokens, resulting in a final dataset of 53K samples.

For these 53K self-refinement problems, we utilize QwQ-32B model and generate up to 10 solutions for each problem in our dataset. We use temperature of 0.7, top-p of 0.95, and limit generations to 16,384 tokens. The generated solutions

Table 6: Step by step filtering process for the construction of problems. The process starts from a large-scale raw dataset and applies a series of filtering operations to yield the final curated dataset. The number of unique problems and their corresponding generated outputs are tracked at each step.

Step	Filtering Operation	Problems	Generations
-	Initial Raw Dataset	540K	3.2M
1	Remove generations not used in Kaggle competition.	-	2.2M
2	Retain data with 'problem_type' == 'has_answer_extracted'.	-	1.3M
3	Discard problems that have no 'pass_rate_72b_tir'.	-	1.2M
4	Aggregate generations by their corresponding unique problem.	116K	1.2M
5	Filter problems to keep only those with a 'pass_rate_72b_tir' between 0.25 and 0.90.	56K	617K

Table 7: Step by step curation of final hybrid training datasets from the prefiltered pool.

Path	Step	Operation	Problems	Generations
-	-	Prefiltered Pool (from Table 6 Step 5).	56K	617K
$\mathcal{D}_{selfR}$	1	Use all 56K problems for self-refinement generation.	56K	-
	2	Generate 6 solutions as candidates using Qwen2.5-7B-Instruct.	56K	336K
	3	Construct augmented prompts for self-refinement.	56K	-
	4	Remove data where the prompt length exceeded 8,192 tokens.	53K	-
	5	Generate 10 solutions per problem using QwQ-32B.	53K	530K
	6	Subsample final solutions based on correctness.	-	184K
$\mathcal{D}_{direct}$	1	Randomly sample 184K generations from the pre-filtered pool.	-	184K
Final Dataset	1	Merge the direct-answer dataset with self-refinement dataset.	-	368K

by QwQ-32B are then filtered based only on the correctness. Specifically, if a problem yields between one and nine correct solutions (out of 10), we retain all of them. However, for problems where all 10 generated solutions are correct, we randomly sample four of them. This curation process yields our final training dataset $\mathcal{D}_{selfR}$ for self-refinement task of 184K samples.

For the direct-answer dataset, we randomly sample an equal number of 184K, the size of our self-refinement dataset, from the dataset obtained after Step 5 in Table 6 to create our $\mathcal{D}_{direct}$ dataset. We then merge the self-refinement dataset $\mathcal{D}_{selfR}$ with direct-answer dataset $\mathcal{D}_{direct}$ to create our final hybrid 368K training dataset.

F Case Study

In this section, we present a more detailed case study from AIME24. This case demonstrates that our method can recover from four incorrect candidate responses and produce a final correct solution.

F.1 Original Question and Candidate Solutions

The following is the original problem and four candidate responses provided to the model. Due to space constraints, the content of the candidate responses has been condensed.

Question and Candidate Solutions

You are an expert and creative solver, provided with a challenging problem and a set of candidate responses which may be correct, partially correct or even wrong.
You should first fully summarize the connection between candidate responses and problem, then generate a new and superior solution. You should generate a correct solution yourself if all candidates are wrong. Don't copy candidates, use insights selectively and reason independently.
Problem:
Let $\triangle ABC$ have circumcenter O and incenter I with $\overline{IA} \perp \overline{OI}$, circumradius 13, and inradius 6. Find $AB \cdot AC$. Let's think step by step and output the final answer within `\boxed{}`. Candidate Response 1:
...Given the symmetry and the conditions, we can use the known result for the product of the sides in a right triangle:

$$AB \cdot AC = 384$$

Thus, the product $AB \cdot AC$ is:

$$\boxed{384}$$

Candidate Response 2:
...Thus, the final answer is

$$\boxed{26\sqrt{39}}$$

Candidate Response 3:
...Thus, the final answer is:

$$\boxed{312}$$

Candidate Response 4:
...Thus, the final answer is:

$$\boxed{26}$$

F.2 Thinking Process of Self-Refinement

In this section, we present a more detailed trace of the model's thinking component and summary component, when provided with a problem and four candidate responses. We split and structure the model's raw thinking process into four parts, each marked by **a bold title**. To provide analytical insights, we have inserted *meta comment in blue italics*, focusing on how the model analyzes, discerns, and selectively utilizes valuable information from four candidates responses. We give special attention to the model's ability to self-correct by analyzing candidate responses, identifying flaws and choosing an alternative decoding path. To make the model's direct engagement explicit, its verbatim references to the candidate responses are highlighted in red.

Thinking component of GSR-7B

1. Initial analysis and calculation (Consistent with candidates)
Okay, let me try to tackle this problem step by step. ... First, I remember that there's a formula connecting the distance between the incenter and circumcenter (OI) with the circumradius (R) and inradius (r). The formula is $OI^2 = R(R - 2r)$. Let me plug in the given values: $R = 13$ and $r = 6$. So, $OI^2 = 13(13 - 12) = 13 \times 1 = 13$, so $OI = \sqrt{13}$. That's straightforward. Next, the problem states that IA is perpendicular to OI. Since $IA \perp OI$, triangle OIA is a right triangle with right angle at I. The hypotenuse here is OA, which is the circumradius, so $OA = 13$. Using the Pythagorean theorem on triangle OIA: $OA^2 = OI^2 + IA^2$. Plugging in the known values: $13^2 = (\sqrt{13})^2 + IA^2 \rightarrow 169 = 13 + IA^2 \rightarrow IA^2 = 156 \rightarrow IA = \sqrt{156}$. *[Meta comment: The model's initial calculations align perfectly with the correct steps found in Candidates 1 and 2, thereby establishing a valid foundational starting point. The model also notes a critical flaw in Candidates 3 and 4: they employ an incorrect formula, $OI^2 = R^2 - 2Rr$. Although this formula coincidentally produces the correct numerical value for the specific given radius $R = 13$, its structural inaccuracy leads the model to correctly flag these candidates as unreliable early in the process.]*

2. Deep dive and critique of candidate strategies
[Meta comment on Candidate 1: The model observes that Candidate 1's final answer, 384, is presented without a clear derivation. The model formulates a hypothesis: this answer might originate from an unstated and potentially incorrect assumption that the triangle is right-angled at vertex A. It then proceeds to rigorously test this hypothesis.]

Alternatively, perhaps the triangle is right-angled? If $\triangle ABC$ is right-angled, then the circumradius would be half the hypotenuse. Since the circumradius is 13, then the hypotenuse would be 26. Suppose angle A is 90°, then BC would be the hypotenuse, so $BC = 26$. Then, using the inradius formula: $r = (AB + AC - BC)/2$. Given $r = 6$, so $(AB + AC - 26)/2 = 6 \rightarrow AB + AC = 38$. Also, by Pythagoras, $AB^2 + AC^2 = 26^2 = 676$. ... $2AB \cdot AC = 1444 - 676 = 768 \rightarrow AB \cdot AC = 384$. Wait, that's one of the candidate answers (Candidate 1's answer is 384). But does this satisfy the condition $IA \perp OI$? ... Let me set coordinates: Let's place A at $(0, 0)$, B at $(0, b)$, C at $(c, 0)$. ... in a right-angled triangle at A(0,0), with legs on axes, ... the incenter is at (r, r). So here, I would be at $(6, 6)$. The circumcenter O is the midpoint of BC, which is at $(c/2, b/2)$. Now, we need to check if IA is perpendicular to OI. Vector $\vec{IA}$ is from I(6,6) to A(0,0): $(-6, -6)$.

Vector $\vec{OI}$ is from O(c/2, b/2) to I(6,6): $(6-c/2, 6-b/2)$. Their dot product should be zero: $(-6)(6-c/2)+(-6)(6-b/2) = 0 \rightarrow -36+3c/2-36+3b/2 = 0 \rightarrow -72+(3/2)(b+c) = 0$. But since $b+c = 38, \rightarrow -72+(3/2)*38 = -72+57 = -15 \neq 0$. Thus, the dot product is not zero, so the condition $IA \perp OI$ is not satisfied. ...but the calculation gave AB·AC=384, which is the first candidate's answer, but it doesn't satisfy the perpendicular condition. So that must be incorrect... Therefore, the triangle cannot be right-angled at A. Hmm, so that approach was wrong.

[Meta comment on Candidate 2: The model shifts its analysis to the other candidates and identifies a valuable, correct intermediate step within Candidate 2's reasoning. It decides to pursue this line of reasoning to evaluate its potential and identify where, if at all, it deviates from a correct solution path.]

...Wait, another formula for IA is $IA = 2R\sin(A/2)$. Let me check that. Yes, I think that's correct. ... $IA = 2R\sin(A/2) \rightarrow \sqrt{156} = 2 \cdot 13 \cdot \sin(A/2) \rightarrow \sin(A/2) = \sqrt{156}/(26) = \sqrt{39}/13$. ... Thus, $\sin A = 2\sin(A/2)\cos(A/2) = 2 \cdot (\sqrt{39}/13) \cdot (\sqrt{130}/13) = \cdots = (2\sqrt{30})/13$. ... We had $\sin A = 2\sqrt{30}/13$. Then $\cos A = \sqrt{1-\sin^2 A} = \sqrt{1-(4 \cdot 30)/169} = \sqrt{(169-120)/169} = \sqrt{49/169} = 7/13$.

[Meta comment: The model confirms the validity of Candidate 2's calculations regarding the trigonometric properties of angle A, acknowledging this as a key insight. However, it also diagnoses the critical failure point in Candidate 2's approach: the strategy becomes untenable when it hinges on calculating the semiperimeter, 's', a step where the candidate's method falters. The model elects to adopt the correct trigonometric values derived by Candidate 2 while consciously avoiding the flawed semiperimeter-dependent strategy.]

...So, $6s = (1/2) \cdot AB \cdot AC \cdot (2\sqrt{30}/13) \rightarrow 6s = (AB \cdot AC \cdot \sqrt{30})/13 \rightarrow AB \cdot AC = (6s \cdot 13)/\sqrt{30}$. Hmm, so if I can find s, then I can compute $AB \cdot AC$... This is the path Candidate 2 took, but they failed to find s correctly. I will retain the trigonometric values for angle A but seek a more direct path to the solution, avoiding this semiperimeter dead end.

3. Synthesizing a novel, correct solution from useful parts

[Meta comment: After systematically identifying the flaws in Candidates 1, 3, and 4, and the dead-end in Candidate 2's approach, the model determines that a novel solution path is required. The chosen strategy is a synthesis: it leverages the correct trigonometric values for angle A, which were validated from Candidate 2's work, and integrates them with a powerful trigonometric identity not utilized by any of the candidates.]

...We have $r = 6, R = 13$, so: $6 = 4 \cdot 13 \cdot \sin(A/2)\sin(B/2)\sin(C/2)$. We already found $\sin(A/2) = \sqrt{39}/13$. Let me plug that in: $6 = 52 \cdot (\sqrt{39}/13) \cdot \sin(B/2)\sin(C/2) \rightarrow 6 = 4\sqrt{39} \cdot \sin(B/2)\sin(C/2) \rightarrow \sin(B/2)\sin(C/2) = 6/(4\sqrt{39}) = 3/(2\sqrt{39})$. ... Also, since $B+C = \pi - A$, $\sin(B/2)\sin(C/2) = [\cos((B-C)/2) - \cos((B+C)/2)]/2 = [\cos((B-C)/2) - \sin(A/2)]/2$. So, $[\cos((B-C)/2) - \sqrt{39}/13]/2 = 3/(2\sqrt{39}) \rightarrow \cos((B-C)/2) - \sqrt{39}/13 = 3/\sqrt{39} \rightarrow \cos((B-C)/2) = 3/\sqrt{39} + \sqrt{39}/13 = \cdots = 2\sqrt{39}/13$. ... Now, using the identity for $\sin B \sin C$: $\sin B \sin C = [\cos(B-C) - \cos(B+C)]/2$. We have $\cos(B-C) = 2\cos^2((B-C)/2) - 1 = 2 \cdot ((2\sqrt{39}/13)^2) - 1 = \cdots = 143/169$. And $\cos(B+C) = -\cos A = -7/13$. $\sin B \sin C = [143/169 - (-7/13)]/2 = [143/169 + 91/169]/2 = (234/169)/2 = 117/169$. ... Thus, $AB \cdot AC = 4R^2 \sin B \sin C = 4 \cdot (13)^2 \cdot (117/169) = 4 \cdot 169 \cdot (117/169) = 4 \cdot 117 = 468$.

4. Final verification of the self-derived answer

...Hmm, so the correct answer should be 468. But none of the candidates have this. ... let me see if that makes sense with the area. The area $\Delta = (AB \cdot AC \sin A)/2 = (468 \cdot (2\sqrt{30}/13))/2 = (468 \cdot \sqrt{30})/13 = 36\sqrt{30}$. Also, the area is $rs = 6s$. Thus, $6s = 36\sqrt{30} \rightarrow s = 6\sqrt{30}$. ... The sides are $a = 4\sqrt{30}$, and $b + c = 2s - a = 12\sqrt{30} - 4\sqrt{30} = 8\sqrt{30}$. Thus, $b + c = 8\sqrt{30}$, and $bc = 468$. ... We can check if these satisfy the Law of Cosines: $\cos A = (b^2+c^2-a^2)/(2bc)$. We have $\cos A = 7/13$. ... $b^2 + c^2 = (b + c)^2 - 2bc = (8\sqrt{30})^2 - 2(468) = 1920 - 936 = 984$. Plugging into Law of Cosines: $(984 - 480)/(2 \cdot 468) = 504/936 = 7/13$. This matches the earlier result. Thus, the calculations are consistent. Therefore, the correct answer should be 468.

F.3 Summary Component of Self-Refinement

We now present the summary component generated by the model. This component serves as a concise distillation of the core content detailed in the preceding chain-of-thought analysis. It is worthy that the model autonomously provides a novel solution path, distinct from all candidate responses, to arrive at the correct answer of **468**.

Summary component of GSR-7B

To solve the problem, we need to find the product $AB \cdot AC$ for triangle ABC with circumradius $R = 13$, inradius $r = 6$, and the condition that $\overline{IA} \perp \overline{OI}$.

Key Steps:

1. **Distance Between Incenter and Circumcenter**: Using Euler's formula:

$$OI^2 = R(R - 2r) = 13(13 - 12) = 13$$
$$\implies OI = \sqrt{13}$$

2. **Right Triangle Condition**: Since $\overline{IA} \perp \overline{OI}$, triangle OIA is right-angled at I. Using the Pythagorean theorem:

$$OA^2 = OI^2 + IA^2 \implies 13^2 = (\sqrt{13})^2 + IA^2 \implies IA^2 = 156 \implies IA = \sqrt{156}$$

3. **Trigonometric Relationships**: - The distance IA can also be expressed as $IA = 2R \sin\left(\frac{A}{2}\right)$:

$$\sqrt{156} = 2 \cdot 13 \cdot \sin\left(\frac{A}{2}\right)$$
$$\implies \sin\left(\frac{A}{2}\right) = \frac{\sqrt{39}}{13}$$

- Using the double-angle identity for $\sin A$:

$$\sin A = 2 \sin\left(\frac{A}{2}\right) \cos\left(\frac{A}{2}\right)$$
$$= 2 \cdot \frac{\sqrt{39}}{13} \cdot \sqrt{\frac{130}{169}} = \frac{2\sqrt{30}}{13}$$

- The cosine of angle A is:

$$\cos A = \sqrt{1 - \sin^2 A} = \frac{7}{13}$$

4. **Product of Sides Using Law of Sines**:
- The sides AB and AC are given by:

$$AB = 2R \sin C \quad \text{and} \quad AC = 2R \sin B$$

- The product $AB \cdot AC$ is:

$$AB \cdot AC = 4R^2 \sin B \sin C$$

- Using trigonometric identities and the given conditions, we find:

$$\sin B \sin C = \frac{117}{169}$$

- Substituting $R = 13$ and $\sin B \sin C$:

$$AB \cdot AC = 4 \cdot 13^2 \cdot \frac{117}{169} = 4 \cdot 117 = 468$$

Final Answer:

$$\boxed{468}$$