

Double-Checker: Enhancing Reasoning of Slow-Thinking LLMs via Self-Critical Fine-Tuning

Xin Xu^{1*}, Tianhao Chen^{1*}, Fan Zhang¹, Wanlong Liu², Pengxiang Li³,
Ajay Kumar Jaiswal⁴, Yuchen Yan⁵, Jishan Hu¹, Yang Wang¹, Hao Chen¹, Shiwei Liu⁶,
Shizhe Diao⁷, Can Yang^{1†}, Lu Yin^{8†}

¹ The Hong Kong University of Science and Technology

² University of Electronic Science and Technology of China

³ Dalian University of Technology ⁴ University of Texas at Austin ⁵ Zhejiang University

⁶ University of Oxford ⁷ NVIDIA ⁸ University of Surrey

{xxuca, tchenbb}@connect.ust.hk, macyang@ust.hk, l.yin@surrey.ac.uk

Abstract

While slow-thinking large language models (LLMs) exhibit reflection-like reasoning, commonly referred to as the “aha moment”, their ability to generate informative critiques and refine prior solutions remains limited. In this paper, we introduce *Double-Checker*, a principled framework designed to enhance the reasoning capabilities of slow-thinking LLMs by fostering explicit self-critique and iterative refinement of their previous solutions. By fine-tuning on our curated 1,730 self-critical instances, *Double-Checker* empowers long-CoT LLMs to iteratively critique and refine their outputs during inference until they evaluate their solutions as correct under self-generated critiques. We validate the efficacy of *Double-Checker* across a comprehensive suite of reasoning benchmarks, demonstrating that iterative self-critique significantly enhances the reasoning capabilities of long-CoT LLMs. Notably, our *Double-Checker* increases the pass@1 performance on challenging AIME benchmarks from 4.4% to 18.2% compared to the original long-CoT LLMs. These results highlight a promising direction for developing more trustworthy and effective LLMs capable of structured self-critique. Our codes and data are available at <https://github.com/XinXU-USTC/DoubleChecker>.

1 Introduction

Reasoning—the capacity to solve complex tasks by logically connecting facts and drawing conclusions—represents a critical milestone in the quest for Artificial General Intelligence [1, 2, 3, 4]. Following the advent of large language models (LLMs), extensive research has sought to further enhance their reasoning ability, spanning more effective pretraining [5, 6], supervised fine-tuning [7, 8, 9, 10, 11], rigorous evaluation [4, 12, 13], and, more recently, reinforcement learning (RL) [14, 15, 16]. In particular, [15] shows that RL with verifiable rewards can push LLMs toward generating *long chains of thought* (long-CoT) [17] and exhibiting reflective-like reasoning behavior, often termed as the “aha moment” [18]. Despite these gains, Recent works [19, 20] suggest that revisiting and refining previous solutions might unlock further improvements, motivating us to integrate the “aha moment” into a systematic “reflect-and-refine” loop.

The key concept lies in the “reflect-and-refine” is *critique*: the model’s explicit evaluation of whether a solution is correct and, if needed, how it can be improved [21, 22, 23]. Critique underpins the principle of selectively refining only those solutions that need fixing, thereby preserving originally correct

* Equal contribution.

† Corresponding author.

<p>Question: Among the 900 residents of Aimeville, there are 195 who own a diamond ring, 367 who own a set of golf clubs, and 562 who own a garden spade. In addition, each of the 900 residents owns a bag of candy hearts. There are 437 residents who own exactly two of these things, and 234 residents who own exactly three of these things. Find the number of residents of Aimeville who own all four of these things. Answer: 73</p>		
<p>DeepSeek-Distill-Qwen-7B</p> <pre><think> </think> Given the hyperbola, ... Thus, the final answer is [480] + probe prompt to induce self-critique: Alright, let's go through the solution step by step ... Therefore, the final answer is [480]</pre>	<p>DoubleChecker-DS-7B Round 0</p> <pre><think> </think> <summary> Given the hyperbola, ... Thus, the final answer is [480] </summary></pre>	<p>DoubleChecker-DS-7B Round 1</p> <pre><critique> Overall Judgment: Incorrect </critique> <think> </think> <summary>, the correct answer should be [73] </summary></pre>

Figure 1: Double-Checker correctly solves a math problem in AIME24 leveraging self-critique, while DeepSeek-Qwen-7B still gets the same wrong answer under a self-critical probe.

answers [24, 25]. Numerous studies have demonstrated that critique can subsequently be utilized to enhance the quality of generated outputs [22, 23, 26]. For example, [22, 27] train specialized critique-oriented LLMs capable of providing feedback to generator LLMs. However, employing a separate model exclusively for critique introduces additional overhead [22]. Alternatively, [23] proposes integrating critique as a training objective. Nevertheless, the resulting LLMs are unable to leverage self-critique effectively during inference. Furthermore, [19] reports only marginal improvements for Long-CoT LLMs, even after fine-tuning on 100K self-critique examples. These raise an open question: *Do Long-CoT LLMs, which demonstrate reflection-like reasoning, possess the capacity to leverage self-critique to enhance performance? If not, how can we equip them with this ability?*

In this paper, we investigate the integration of reflection-like reasoning with self-critique to enhance the reasoning abilities of slow-thinking LLMs. Specifically, we start by examining whether long-CoT LLMs can leverage self-critique to iteratively refine their prior solutions during inference in a probe-induced manner. Our findings reveal that the occurrence of an "aha moment" does not necessarily indicate the presence of a self-critique mechanism (see Sec. 3.1). For instance, as illustrated in Fig. 1, DeepSeek-Distill-Qwen-7B fails to generate informative critiques of its prior solution, ultimately arriving at the same incorrect answer. To address this, we introduce Double-Checker, a novel framework designed to empower LLMs to critique and refine their prior solutions iteratively and adaptively. Through a specialized training process that combines direct inference instances with curated critique-refine data (1,730 instances in total), our Double-Checker equips long-CoT LLMs with an effective self-critique capability. This enables iterative improvements in performance during inference via self-critique. An example of this process is shown in Fig. 1, where Double-Checker successfully resolves a complex math problem using the "reflect-and-refine" approach.

Our main contributions can be summarized as follows: ❶ We investigate the self-critical behavior of long-CoT LLMs via a probing and find that they are unable to generate informative critiques to improve their prior solutions. ❷ We propose Double-Checker, a novel framework that pairs direct inference data with a carefully curated critique-refine dataset (1,730 in total), enabling LLMs to *iteratively* correct flawed reasoning during inference. ❸ Experiments on a wide range of reasoning benchmarks demonstrate that even with a modest amount of critique data, Double-Checker unlocks substantial improvements in accuracy. Notably, our method raises pass@1 performance on challenging AIME benchmarks from 4.4% to 18.2%, underscoring the impact of explicit self-critique.

2 Related Work

Long Chain-of-Thought and Slow Thinking. The rise of LLMs has driven extensive research to enhance their reasoning capabilities through various strategies. Early efforts include advancements in pretraining methodologies [5, 6], supervised fine-tuning [7, 8, 9, 10, 11], and rigorous evaluation techniques [4, 12, 13]. More recently, RL has emerged as a key paradigm for improving reasoning in LLMs. For instance, [15] demonstrates that RL with verifiable rewards enables models to generate

long chains of thought (long-CoT) [17], fostering more structured, multi-step problem-solving skills. This approach has been shown to promote reflective reasoning behaviors, termed as the "aha moments" [18]. These advancements mark significant progress in LLM reasoning [28]. However, our work reveals a critical limitation: while strong reflection-like reasoning allows LLMs to recognize errors or inconsistencies, it does not inherently ensure robust self-improvement [29, 30]. Additionally, existing self-improvement methods [29, 30] often depend on external tools or explicit feedback mechanisms, making it challenging to guide a single LLM through multiple, reliable rounds of refinement. Addressing these challenges is crucial to unlocking the full potential of self-improvement in LLMs.

Critique LLMs and Integrated Self-Improvement. A parallel line of research employs *critique models* or reward estimators to score and refine outputs from a "generator" model, especially in mathematical domains [31, 32, 33, 34]. While effective in principle, this split-architecture strategy requires substantial overhead (running two separate LLMs) or produces numeric feedback that lacks actionable corrections [35]. Other efforts have tried to incorporate critique into a single model's training objective [23] or train on large multi-round self-critique data [19], but with limited gains in *iterative* refinement. Against this backdrop, our work introduces Double-Checker, which merges critique and generation into a unified "reflect-and-refine" loop within *one* long-CoT LLM. By carefully curating critique-oriented examples and integrating them with direct-inference data, we equip long-CoT LLMs with the capability to generate meaningful critiques and adaptively refine their prior solutions based on self-generated critiques, ultimately enabling robust self-improvement.

3 Method

3.1 Aha Moment Does Not Equate to Effective Self-Critique

Previous studies have observed that fast-thinking LLMs often generate uninformative critiques, limiting their capacity for self-improvement [22, 36]. In contrast, slow-thinking LLMs are believed to exhibit self-reflection behaviors, identifying and potentially correcting errors in their reasoning steps [15]. This raises the intriguing question: Can long-CoT LLMs with strong reflection-like reasoning abilities perform effective self-critique? To investigate this, we conduct experiments on AIME24 using DeepSeek-R1-Distill-7B and DeepSeek-R1-Distill-32B, employing a probe to induce self-critique behavior (see Appendix A.1 for detailed settings). We have the following results: ❶ DeepSeek-R1-Distill-7B and DeepSeek-R1-Distill-32B follow the probe prompt and produce informative critiques in only 0% and 8.5% of cases, respectively. ❷ The performance on AIME24 improves slightly after refinement with self-critique (1.6% for 7B: 57.1% \rightarrow 58.7% and 0.8% for 32B: 72.1% \rightarrow 72.9%). These findings suggest that the aha moment does not inherently translate into effective self-critique. While these models demonstrate strong capabilities in reflection-type reasoning, their capacity to autonomously evolve through effective self-critique remains limited.

3.2 Double-Checker Framework

Despite exhibiting the "aha moment," long-CoT LLMs demonstrate limited ability to generate actionable critiques and effectively apply them for iterative self-refinement. We hypothesize that this limitation arises because current long-CoT LLMs are primarily trained for direct inference. As a result, these models do not naturally transition toward interactive refinement through self-critique, even when prompted with carefully designed probes (see Sec. 3.1). To address this gap, we propose Double-Checker, a novel framework designed to enable long-CoT LLMs to critique their prior solutions and iteratively refine their reasoning. An overview of Double-Checker is depicted in Fig. 2. This section presents the detailed training and inference process of Double-Checker.

3.2.1 Training Process

As shown in Fig. 2 (c), the training process of Double-Checker consists of four key steps: 1) Initial Generation, 2) Critique with Answer Correctness, 3) Refinement, 4) Distillation. The first three steps focus on data curation, while the final step involves model training. Start from an original training dataset $\mathcal{D}_{orig} = \{(Q_i, GT_i)\}$, which consists of multiple questions with their corresponding ground-truth answers, we will detail each step below.

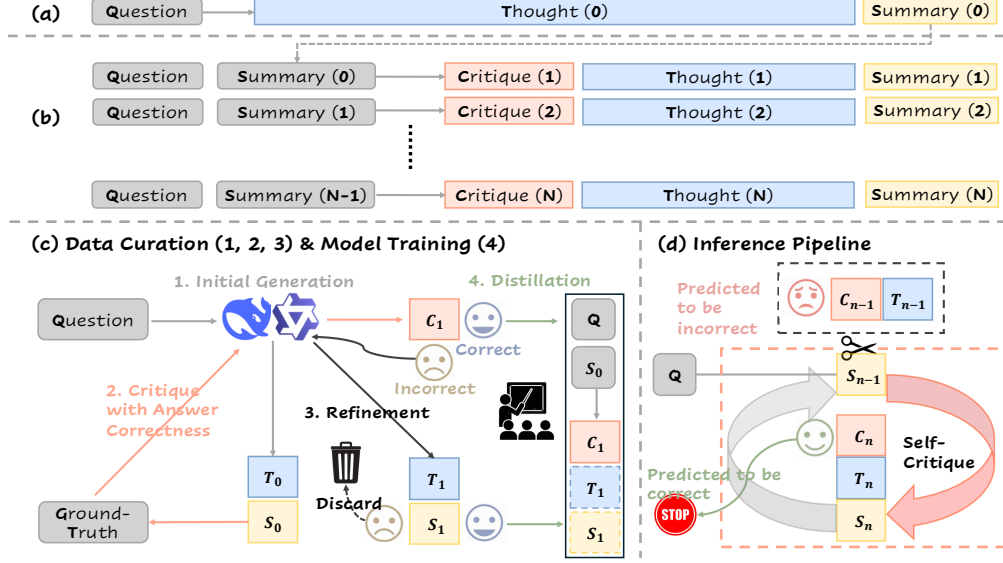


Figure 2: The overview of Double-Checker. (a) Direct inference pipeline of long-CoT LLMs: generating a long thought (T_0) followed by a summary (S_0) that concludes the answer (A_0) for the question (Q). (b) The inference pipeline of iterative refinement with self-critique. (c) Training stage of our Double-Checker. (d) Adaptive inference with self-critique of our Double-Checker.

Initial Generation. For each question Q , we first obtain its direct inference result from a strong teacher long-CoT LLM \mathcal{T} (e.g., DeepSeek-R1) as: $\mathcal{T} : Q \rightarrow T_0 \oplus S_0$, where S_0 contains the model-generated initial answer A_0 . The answer A_0 is then evaluated for correctness by comparing it with the ground-truth answer GT .

Critique with Answer Correctness. Given a question Q and its preceding summary S_0 , we employ a proficient LLM \mathcal{C} to generate detailed critiques. The critique explicitly signals the correctness of the initial answer A_0 (correct/incorrect). To optimize critique quality, we employ distinct prompts tailored to correct and incorrect A_0 (see App. A.3). This answer correctness signal is indispensable for effective critique generation. Formally, critique generation follows:

$$\mathcal{C} : \{\text{Instruction incorporating Answer Correctness Signal}\} \oplus Q \oplus S_0 \rightarrow C_1$$

where C_1 adheres to the structured format defined in App. A.2.

Refinement. When the initial answer A_0 is correct, we collect the corresponding critique C_1 and store the triplet (Q, S_0, C_1) into our training set $D_{critique}$. For incorrect A_0 , we refine the solution using a Refinement long-CoT LLM \mathcal{R} , which takes the question Q , prior summary S_0 , and critique C_1 as input: $\mathcal{R} : Q \oplus S_0 \oplus C_1 \rightarrow T_1 \oplus S_1$, where T_1 and S_1 represent the refined reasoning and summary, respectively. The refined answer A_1 is extracted from S_1 and compared to the ground truth GT . If A_1 matches GT , (Q, S_0, C_1, T_1, S_1) is added to $D_{critique}$; otherwise, it is discarded.

Distillation. After the data curation stage, training examples are categorized into two formats: (Q, S_0, C_1) for correct answers A_0 and (Q, S_0, C_1, T_1, S_1) for incorrect answers. For simplicity, instances with (Q, S_0, C_1) are padded to $(Q, S_0, C_1, T'_1, S'_1)$ using a predefined template (see App. A.4). To maintain the ability of direct inference of the original long-CoT LLM \mathcal{M} , we will mix $D_{critique}$ with a direct inference training set $D_{direc} = \{(Q, T_0, S_0)\}$. Finally, our training set will be $D_{train} = D_{direc} \cup D_{critique}$. The learning objective is

$$\min_{\theta} \left\{ -\frac{1}{|D_{train}|} \left(\sum_{D_{direc}} \log \mathbb{P}_{\mathcal{M}_{\theta}}(T_0 \oplus S_0 | Q) + \sum_{D_{critique}} \log \mathbb{P}_{\mathcal{M}_{\theta}}(C_1 \oplus T_1 \oplus S_1 | Q \oplus S_0) \right) \right\},$$

where θ is the parameters of long-CoT LLM \mathcal{M} and $|D_{train}|$ denotes the number of training examples.

3.2.2 Inference Pipeline

We will first introduce a paradigm shift from direct inference (Fig. 2 (a)) to iterative refinement via self-critique (Fig. 2 (b)). Concretely:

Algorithm 1 Double-Checker Inference Pipeline

Require: Question Q , Long-CoT LLM \mathcal{M} , number of iterations N

```
1: Generate initial output  $T_0 \oplus S_0 \sim \mathbb{P}_{\mathcal{M}}(\cdot|Q)$   $\triangleright$ Direct Inference
2: for  $n \leftarrow 1$  to  $N$  do
3:   Critique previous summary and refine  $C_n \oplus T_n \oplus S_n \sim \mathbb{P}_{\mathcal{M}}(\cdot|Q \oplus S_{n-1})$   $\triangleright$ Self-Critique & Refine
4:   if  $C_n$  indicates that  $A_{n-1}$  (the answer of  $S_{n-1}$ ) is correct then  $\triangleright$ Stopping Criteria
5:     return  $S_{n-1}$ 
6:   end if
7: end for
8: return  $S_N$ 
```

- *Round 0 (Direct Inference).* Given a question Q , the model \mathcal{M} generates a detailed reasoning chain T_0 and a final summary S_0 , i.e.,

$$\mathcal{M} : Q \rightarrow T_0 \oplus S_0.$$

where \oplus denotes the string concatenation. This baseline (long-CoT) forms the initial solution.

- *Round 1 (Self-Critique + Refinement).* We now feed both Q and the prior summary S_0 to \mathcal{M} . The model produces a critique C_1 of S_0 and then refines the solution into a new thought T_1 , finally yielding a new summary S_1 . Formally,

$$\mathcal{M} : Q \oplus S_0 \rightarrow C_1 \oplus T_1 \oplus S_1.$$

- *Round n (Repeated Refinement).* For subsequent rounds ($1 \leq n \leq N$), the model receives $Q \oplus S_{n-1}$, generates C_n to critique the previous summary, and refines the solution into T_n and S_n . Symbolically,

$$\mathcal{M} : Q \oplus S_{n-1} \rightarrow C_n \oplus T_n \oplus S_n.$$

We continue until the critique C_n deems the answer correct or a maximum iteration limit N is reached.

Context Window. The thought T_i is typically lengthy, while the corresponding summary S_i usually encapsulates all the essential information of T_i , serving as a concise version of T_i . Discarding T_i and retaining only S_i for each refinement round will ensure that the entire refinement process remains within the context window of Long-CoT LLMs.

Critique Space. The *critique* evaluates the prior summary, assessing whether the answer is correct and proposing actionable suggestions to enhance the solution when needed. Following [22], our critique consists of three components: 1) an analysis of the summary, 2) actionable improvement suggestions, 3) an answer correctness judgment (correct/incorrect). This judgment enables early termination of the iterative refinement process when the solution is deemed correct (see Sec. 3.2). An example of the critique structure is provided in Appendix A.2.

As illustrated in Fig. 2 (d), our Double-Checker adopts an iterative refinement pipeline that alternates between: (1) appending the previous summary (S_{n-1}) to the input question (Q), and (2) generating an informative critique (C_n) followed by refining the prior solution (T_n, S_n). The process terminates when the critique C_n predicts the correctness of the answer extracted from S_{n-1} . The complete inference procedure is formally presented in Algorithm 1. To ensure termination, we define a maximum iteration limit N . Although theoretically the process could iterate infinitely until all test examples achieve critique-verified correctness, we impose a finite N in practice to prevent computational divergence due to potential inability to predict correctness for certain cases.

4 Experiments and Results

4.1 Training Setup

Training Data Construction. To construct the original training set \mathcal{D}_{orig} , we compile a pool of candidate problems from existing mathematical reasoning datasets: S1.1 [11], DeepMath-103K [37], OpenRS [38], and ORZ-Math-Hard [39]. We filter these candidates using two key criteria: 1) Answer Verifiability: Ensuring ground-truth labels are verifiable via rule-based validation, 2) Difficulty: Selecting problems with appropriate complexity. We get a collection of around 8K high-quality

questions, calibrated for both difficulty and correctness. For initial generation, critique annotation, and refinement, we utilize Qwen3-235B-A22B [40] and DeepSeek-R1 [15], i.e., $\mathcal{T} = \mathcal{C} = \mathcal{R}$, but with different instructions. We also incorporate a subset of S1.1 training instances as our \mathcal{D}_{direct} , resulting in a total training set $D_{train} = D_{direct} \cup D_{critique}$ of 1,730 training instances. The details of our data sources and filtering process are given in App. B.1.

Training Details. We train the Distilled long CoT variants of DeepSeek-R1 (7B and 32B parameters) on our curated training set \mathcal{D}_{train} using full-parameter fine-tuning. The training process employs DeepSpeed ZeRO optimization [41] for efficient memory utilization and FlashAttention2 [42] for accelerated training. Following the implementation in [10], we set the maximum sequence length to 16,384 tokens and adopt a learning rate of 5×10^{-6} . Implementation details are in App. B.2.

4.2 Evaluation Setup

Evaluation Setting. We evaluate on AIME24, AIME25, MATH500 [43], and OlympiadBench [44] for mathematical reasoning, and GPQA [12] for multidisciplinary problems. Following [10], we adopt an unbiased pass@1 metric for AIME24 and AIME25, generating 16 samples with a decoding temperature of 0.6. For the remaining benchmarks, we generate 4 samples per problem to report pass@1. We use vLLM [45] to accelerate inference and set the maximum sequence length to be 32,768 tokens. We set N in Algorithm 1 to 3 for Double-Checker-DS-7B and 1 for Double-Checker-DS-32B. A brief introduction to different benchmarks and detailed evaluation setting can be found in App. B.3.

Baselines. We compare Double-Checker against a comprehensive set of baselines, categorized as follows: 1) DeepSeek-R1-Distill-Qwen Series (7B, 32B): Strong long-CoT LLMs distilled from DeepSeek-R1 using 800K examples. 2) S1.1 (7B, 32B) [11]: Two CoT LLMs distilled from DeepSeek-R1 using 1K high-quality from multiple sources. 3) LIM0-32B [10]: A powerful LLM trained on 837 carefully curated examples. 4) InfTyThink (7B, 32B) [46]: Models trained on 333K examples adapted from OpenR1-Math, with results from the original paper using multi-round interactive inference. 5) Light-R1 (7B, 32B) [28]: Two-stage SFT (79K data) + RL-trained models. 6) Naive-SFT Baseline (7B, 32B): DeepSeek-R1-Distill-Qwen trained on questions of our \mathcal{D}_{train} using standard SFT (without critique learning), which can isolate the contribution of training data. 7) We also include OpenAI-o1 series [14] and DeepSeek-R1 for reference. 8) We also adapt ThinkTwice [19] with DeepSeek-R1-Distill-Qwen Series to see the effect of "reflect-and-refine" without any finetuning. 9) Self-Refine: We adopt a similar approach as (author?) [29]. 10) We also introduce a sequential test-time scaling approach by appending "wait" before the end of thinking [11]. For 4) and 7), we report the results from other papers directly (see App. C.1), and run the remaining baselines by our own.

4.3 Main Results

Table 1 summarizes the primary evaluation results on multiple challenging reasoning benchmarks. From Table 1, we have several key observations:

Double-Checker consistently enhances the performance of original long-CoT LLMs across all reasoning benchmarks and model scales. Our model, Double-Checker-DS-7B, outperforms DeepSeek-Distill-Qwen-7B by an average of 4.1%, while Double-Checker-DS-32B exceeds DeepSeek-Distill-32B by 6.6%. Notably, Double-Checker-DS-32B achieves a significant improvement of 18.2% in pass@1 on the AIME25 benchmark, and Double-Checker-DS-7B boosts performance on AIME24 by 9.7%, demonstrating the effectiveness of Double-Checker on complex reasoning tasks.

Self-critique is a crucial factor in driving performance improvements across benchmarks. Compared to the "naive SFT" baseline, which utilizes the same training problems but excludes explicit critical data, our Double-Checker consistently shows superior performance across all benchmarks. On average, Double-Checker outperforms the "naive SFT" baseline by 5.7% for the 7B model and 3.5% for the 32B model. These results underscore the pivotal role of self-critique in enhancing the reasoning of long-CoT LLMs.

Double-Checker demonstrates strong generalizability. Although the training data primarily consists of math reasoning problems, Double-Checker achieves remarkable performance on GPQA, a multidisciplinary QA benchmark. Notably, the only source of reasoning problems from other

Table 1: Main results (in %) on various benchmarks. The best results within each group are in **bold**. * indicates that the results of the corresponding LLM are sourced from their technical reports or other references due to the cost of using APIs or the unavailability of the LLM. Please refer to Appendix C.1 for corresponding references. The remaining results are from our own runs.

Model	AIME24	AIME25	MATH500	Olympiad	GPQA	AVG
OpenAI-o1-Preview*	44.6	37.9	85.5	52.1	73.3	-
OpenAI-o1-mini*	63.6	53.8	90.0	-	60.0	-
OpenAI-o1-1217*	79.2	-	96.4	-	75.7	-
DeepSeek-R1*	79.8	70.0	97.3	-	71.5	-
7B Models						
S1.1-7B	17.5	19.6	80.7	42.8	41.3	40.4
InftyThink-7B*	40.0	-	91.7	-	51.9	-
LightR1-7B-DS	57.1	45.4	90.3	59.0	23.1	55.0
DeepSeek-R1-Distill-7B (initial model)	56.7	43.7	92.2	59.0	35.4	57.4
DeepSeek-R1-Distill-7B + self-refine	55.0	37.5	91.6	58.4	36.8	55.9
DeepSeek-R1-Distill-7B + ThinkTwice	58.7	42.9	92.3	58.7	35.5	57.6
DeepSeek-R1-Distill-7B + Wait	57.5	44.3	93.1	58.8	36.6	58.1
Double-Checker-DS-7B naive SFT	57.1	43.3	91.4	58.6	28.7	55.8
Double-Checker-DS-7B	66.4	48.1	92.7	60.0	40.4	61.5
32B Models						
LIMO-32B	57.1	50.8	93.0	66.1	64.5	66.3
S1.1-32B	56.7	47.5	92.9	58.8	66.8	64.5
InftyThink-32B*	62.5	-	96.0	-	65.6	-
QwQ-32B-Preview	44.2	34.2	89.7	63.6	58.8	58.1
LightR1-32B-DS	76.6	65.4	95.2	67.5	68.7	74.7
DeepSeek-R1-Distill-32B (initial model)	72.1	50.4	93.5	63.0	64.9	68.8
DeepSeek-R1-Distill-32B + ThinkTwice	72.9	54.8	93.5	63.3	64.4	69.8
DeepSeek-R1-Distill-32B + wait	72.8	53.4	94.2	63.8	65.7	70.0
Double-Checker-DS-32B naive SFT	74.6	60.4	93.4	67.2	63.8	71.9
Double-Checker-DS-32B	79.8	68.6	94.3	68.2	66.0	75.4

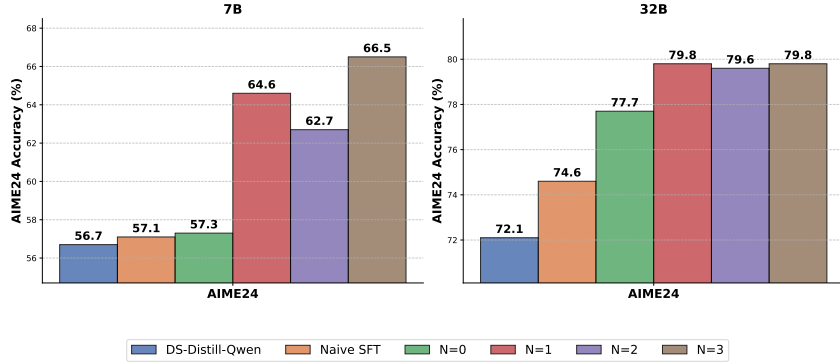


Figure 3: Accuracy comparisons on AIME24 for two model sizes (7B and 32B). We compare: (1) DS-Distill-Qwen (a distilled baseline), (2) Naive SFT (fine-tuning without explicit critique), and (3) Double-Checker with varying rounds of self-critique ($N = 0, 1, 2, 3$).

disciplines is derived from S1.1 dataset, and our training problems constitute a proper subset of S1.1. However, our Double-Checker even shows significant improvements over S1.1, achieving a 21.1% performance gain for the 7B model and a 10.9% gain for the 32B model on GPQA. A similar trend is also observed in LiveCodeBench and MMLU-Pro (Business and Law), further validating the generalizability of Double-Checker, given that it was not trained on any related examples (see App. D.1). Additionally, Double-Checker, which relies solely on SFT, performs comparably to models trained with large-scale RL, such as LightR1. These results align with previous findings that smaller models tend to benefit more from SFT than RL [47, 40].

Table 2: Ablation study (in %) on Training Data.

Model	AIME24	AIME25	MATH500	Olympiad	GPQA	AVG
DeepSeek-R1-Distill-7B	56.7	43.7	92.2	59.0	35.4	57.4
Double-Checker-DS-7B naive SFT	57.1	43.3	91.4	58.6	28.7	55.8
Double-Checker-DS-7B exclude \mathcal{D}_{Direct}	57.5	44.2	88.9	57.3	23.3	54.2
Double-Checker-DS-7B w.o. Qwen3	62.0	45.6	91.6	59.7	39.9	59.8
Double-Checker-DS-7B	66.4	48.1	92.7	60.0	40.4	61.5
Double-Checker-DS-7B + data (N=2)	66.7	46.7	93.2	58.6	40.0	61.1

5 Analysis

5.1 The Effect of Self-Critique

To verify the effectiveness of self-critique, we evaluate different model configurations on AIME24. Figure 3 displays the results for both 7B and 32B models under the following settings:

- *DS-Distill-Qwen*: A distilled long-CoT baseline.
- *Naive SFT*: Fine-tuned with the same problems as our training set without explicit critical data.
- *N=0,1,2,3*: Our Double-Checker approach with $N = 0, 1, 2, 3$ rounds of inference-stage self-critique and refinement.

We notice that at the 7B and 32B scales, self-critique leads to immediate accuracy gains over the distilled baseline. In the 7B setting, moving from $N=0$ (no refinement) to $N=1$ increases performance from 57.3% to 64.6% on AIME24, with further rounds ($N=2,3$) pushing AIME24 up to 66.5%; by contrast, Naive SFT yields only modest improvements over DS-Distill-Qwen. In the 32B setting, Double-Checker starts with a higher performance even at $N=0$ (77.7% on AIME24 vs. 72.1%) and achieves 79.8% on AIME24 by $N=1$, after which performance saturates (79.6%–79.8% on AIME24). This suggests that the 32B model acquires self-critique more rapidly than the 7B model. Additionally, incorporating self-critical data during SFT proves beneficial for enhancing direct inference ability as well ($N = 0$ vs. "naive SFT").

These results confirm the strong positive impact of self-critique. Even a single refinement round ($N=1$) consistently brings notable accuracy gains over baselines, and multiple rounds can yield further improvements—particularly at smaller scales (7B). In contrast, [23] shows even decreased performance of $N = 1$ over $N = 0$ (direct inference). By explicitly learning to critique and update its reasoning, Double-Checker effectively bridges the gap between “long chain-of-thought generation” and “iterative self-improvement”, resulting in strong reasoning power.

5.2 Ablation Study of Training Data

To isolate the contributions of different training data components, we ablate whether (i) we include the original direct-inference data (\mathcal{D}_{Direct}), (ii) we use a naive SFT approach without critique data, and (iii) we exclude Qwen3-annotated examples in our curated dataset, (iv) we further collect training examples that are correctly solved after two rounds of self-critique. Table 2 reports the results on multiple math and reasoning benchmarks.

Excluding Direct Inference Data (exclude \mathcal{D}_{Direct}). Removing the original direct-inference examples (Round 0 data) degrades average accuracy to 54.2%. This decline underscores the importance of retaining a portion of direct inference data in the training mix to preserve the model’s overall reasoning capability. We believe this occurs because training exclusively on $\mathcal{D}_{critique}$ causes the model to lose its ability to perform direct reasoning at $N = 0$.

Effect of removal of Qwen3 Data (w.o. Qwen3). Removing Qwen3-generated critical examples reduces performance from 61.5% to 61.1% on average. Although not as large a drop as excluding direct data, this shows that Qwen3 training instances further enrich the critique set and boost final performance. We believe that scaling up the critical data could yield further performance gains.

Additional Round-2 critique training data (+ data (N = 2)). Adding additional examples that are correctly solved after two rounds of critique during training yields no further improvements. We hypothesize that this is due to the limited sample size (< 400) added, which may be insufficient to influence the training process significantly.

Overall, these results confirm that our curated critique dataset, especially when combined with the original direct-inference examples and auxiliary data from Qwen3, plays a pivotal role in achieving the best performance. In other words, the model benefits from both (i) Conventional long CoT data for maintaining its direct inference ability ($N = 0$) and (ii) explicit self-critique examples for learning to iteratively refine its solutions. (iii) potentially more diverse critical data.

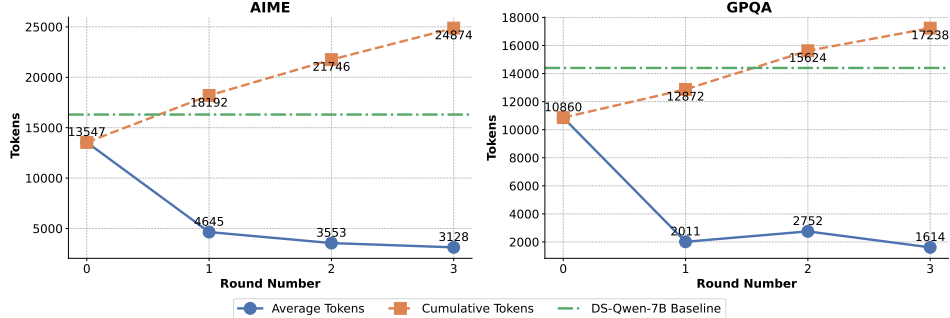


Figure 4: Token usage for AIME24 (left) and GPQA (right). Blue solid line: the per-round average token count, orange dashed line: the cumulative token count over all rounds; green dash-dotted line: the average token consumption for "naive SFT" baseline without iterative refinement.

5.3 Analysis of Response Length

The token usage for different rounds of Double-Checker-DS-7B for AIME24 and GPQA is shown in Fig. 4. We run up to three refinement rounds ($n = 0, 1, 2, 3$) with Double-Checker. We then record both average and cumulative generated tokens used for the full conversation, including thoughts, critiques, and summaries. We also provide the inference time analysis in App. D.2.

On AIME24, the *average tokens per round* drops sharply: from 13.5k at $n = 0$ to 4.6k at $n = 1$, and continues decreasing across subsequent rounds (3.6k at $n = 2$, 3.1k at $n = 3$). A similar pattern holds for GPQA, where the initial 10.9k tokens at $n = 0$ is reduced to roughly 2–3k in the later rounds. Meanwhile, *cumulative tokens* grows steadily with each additional round: for instance, it rises from 13.5k to 24.9k on AIME24 by $n = 3$, and from 10.9k to 17.2k on GPQA. Nevertheless, each new round adds far fewer tokens than the first round. Notably, on GPQA, the total tokens spent in our Double-Checker at $N = 2$ is even smaller than "naive SFT".

Interestingly, the tokens spent on the direct inference of our Double-Checker is fewer than those of "naive SFT" baseline, indicating that incorporating critical data for SFT could also reduce the token consumption at $N = 0$. While allowing more rounds naturally increases the total token count, each round’s contribution is substantially smaller than that of the direct long-CoT baseline. Hence, there is a clear trade-off between improved accuracy (via multiple critique/refine steps) and total token usage. In practice, we find that even a single or two rounds of refinement often suffice to boost correctness without incurring a prohibitive increase in cumulative tokens.

6 Conclusion

We have presented Double-Checker, a framework that explicitly enforces LLMs to critique and refine their previous solutions for self-improvement. Our approach integrates generating reflection-like reasoning and actively correcting potential errors through iterative self-critique. Experimental results on multiple mathematical and multidisciplinary benchmarks demonstrate that Double-Checker consistently improves accuracy over comparable baselines, often by large margins. Furthermore, we emphasize the pivotal role of self-critique during inference in enhancing the reasoning capabilities of long-CoT LLMs. Double-Checker demonstrates the value of equipping LLMs with a structured critique space and training them to reflect and refine their outputs, facilitating more reliable self-improvement for complex reasoning tasks.

Acknowledgments

This work was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project Reference Number: AoE/E-601/24-N).

References

- [1] Edward A Feigenbaum and Julian Feldman. Computers and thought. 1963.
- [2] Meredith Ringel Morris, Jascha Sohl-Dickstein, Noah Fiedel, Tris Warkentin, Allan Dafoe, Aleksandra Faust, Clement Farabet, and Shane Legg. Levels of agi for operationalizing progress on the path to agi. *arXiv preprint arXiv:2311.02462*, 2023.
- [3] Zhen Huang, Zengzhi Wang, Shijie Xia, Xuefeng Li, Haoyang Zou, Ruijie Xu, Run-Ze Fan, Lyumanshan Ye, Ethan Chern, Yixin Ye, et al. Olympicarena: Benchmarking multi-discipline cognitive reasoning for superintelligent ai. *Advances in Neural Information Processing Systems*, 37:19209–19253, 2024.
- [4] Xin Xu, Jiabin Zhang, Tianhao Chen, Zitong Chao, Jishan Hu, and Can Yang. Ugmathbench: A diverse and dynamic benchmark for undergraduate-level mathematical reasoning with large language models. *arXiv preprint arXiv:2501.13766*, 2025.
- [5] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Y Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- [6] An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.
- [7] Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *ArXiv preprint*, abs/2309.12284, 2023.
- [8] Xin Xu, Tong Xiao, Zitong Chao, Zhenya Huang, Can Yang, and Yang Wang. Can llms solve longer math word problems better? *ArXiv preprint*, abs/2405.14804, 2024.
- [9] Yuxuan Tong, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. Dart-math: Difficulty-aware rejection tuning for mathematical problem-solving. *ArXiv preprint*, abs/2407.13690, 2024.
- [10] Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*, 2025.
- [11] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.
- [12] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.
- [13] Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Sean Shi, Michael Choi, Anish Agrawal, Arnav Chopra, et al. Humanity’s last exam. *ArXiv preprint*, abs/2501.14249, 2025.
- [14] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.
- [15] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- [16] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

- [17] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.
- [18] Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*, 2025.
- [19] Xiaoyu Tian, Sitong Zhao, Haotian Wang, Shuaiting Chen, Yunjie Ji, Yiping Peng, Han Zhao, and Xiangang Li. Think twice: Enhancing llm reasoning by scaling multi-round test-time thinking. *arXiv preprint arXiv:2503.19855*, 2025.
- [20] Hasan Abed Al Kader Hammoud, Hani Itani, and Bernard Ghanem. Beyond the last answer: Your reasoning trace uncovers more than you think. *arXiv preprint arXiv:2504.20708*, 2025.
- [21] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-interactive critiquing. *arXiv preprint arXiv:2305.11738*, 2023.
- [22] Zhihui Xie, Liyu Chen, Weichao Mao, Jingjing Xu, Lingpeng Kong, et al. Teaching language models to critique via reinforcement learning. *arXiv preprint arXiv:2502.03492*, 2025.
- [23] Yubo Wang, Xiang Yue, and Wenhui Chen. Critique fine-tuning: Learning to critique is more effective than learning to imitate. *arXiv preprint arXiv:2501.17703*, 2025.
- [24] Ruochen Zhao, Xingxuan Li, Shafiq Joty, Chengwei Qin, and Lidong Bing. Verify-and-edit: A knowledge-enhanced chain-of-thought framework. *arXiv preprint arXiv:2305.03268*, 2023.
- [25] Xin Xu, Shizhe Diao, Can Yang, and Yang Wang. Can we verify step by step for incorrect answer detection? *arXiv preprint arXiv:2402.10528*, 2024.
- [26] Yansi Li, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Qiuzhi Liu, Rui Wang, Zhuosheng Zhang, Zhaopeng Tu, Haitao Mi, et al. Dancing with critiques: Enhancing llm reasoning with stepwise natural language self-critique. *arXiv preprint arXiv:2503.17363*, 2025.
- [27] Wenkai Yang, Jingwen Chen, Yankai Lin, and Ji-Rong Wen. Deepcritic: Deliberate critique with large language models. *arXiv preprint arXiv:2505.00662*, 2025.
- [28] Liang Wen, Yunke Cai, Fenrui Xiao, Xin He, Qi An, Zhenyu Duan, Yimin Du, Junchen Liu, Lifu Tang, Xiaowei Lv, et al. Light-rl: Curriculum sft, dpo and rl for long cot from scratch and beyond. *arXiv preprint arXiv:2503.10460*, 2025.
- [29] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2024.
- [30] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.
- [31] Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.
- [32] An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.
- [33] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, 2024.

- [34] Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels. *arXiv preprint arXiv:2412.01981*, 2024.
- [35] Zachary Ankner, Mansheej Paul, Brandon Cui, Jonathan Daniel Chang, and Prithviraj Ammanabrolu. Critique-out-loud reward models. In *Pluralistic Alignment Workshop at NeurIPS 2024*.
- [36] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*, 2023.
- [37] Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, et al. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*, 2025.
- [38] Quy-Anh Dang and Chris Ngo. Reinforcement learning for reasoning in small llms: What works and what doesn’t. *arXiv preprint arXiv:2503.16219*, 2025.
- [39] Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*, 2025.
- [40] Qwen Team. Qwen3, April 2025.
- [41] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2020.
- [42] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.
- [43] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.
- [44] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024.
- [45] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023.
- [46] Yuchen Yan, Yongliang Shen, Yang Liu, Jin Jiang, Mengdi Zhang, Jian Shao, and Yueting Zhuang. Infythink: Breaking the length limits of long-context reasoning in large language models. *2503.06692*, 2025.
- [47] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.
- [48] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.
- [49] Yixin Ye, Yang Xiao, Tiantian Mi, and Pengfei Liu. Aime-preview: A rigorous and immediate evaluation framework for advanced mathematical reasoning. <https://github.com/GAIR-NLP/AIME-Preview>, 2025. GitHub repository.
- [50] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

- [51] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhui Chen. MMLU-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

A Detailed Settings of Double-Checker

A.1 Setting of Meta-Experiments

For the experiment in Sec. 3.1, we use the following probe to induce self-critique ability, which is adapted from [23]. The evaluation setting on AIME24 aligns with our main experiments (see App. B.3).

To evaluate whether the long CoT LLMs can generate informative critiques using the probe, we examine whether their generated content attempts to assess the correctness of their prior solution. If they do not engage in this answer judgment, we consider the critique to be informative.

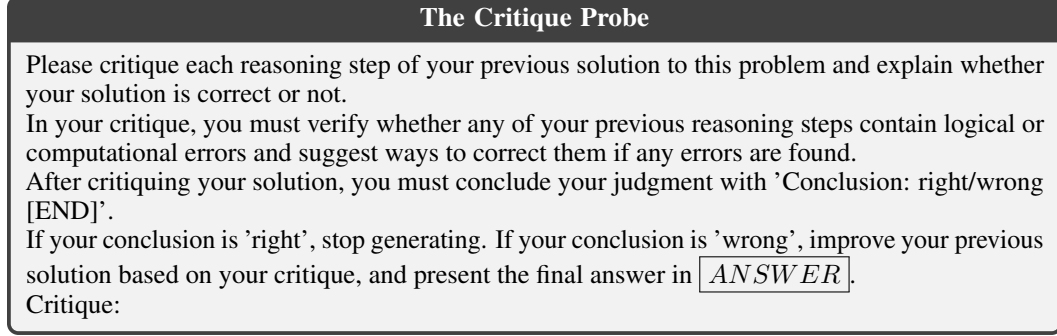


Figure 5: The Probe of Our Experiment in Sec. 3.1.

A.2 The Critique Space

Our critique consists of three components: 1) an analysis of the summary, 2) actionable improvement suggestions, 3) an answer correctness judgment (correct/incorrect).

We showcase one such example in Fig. 6.

A.3 Critique Generation

To ensure the quality of critiques, we employ different prompts for correct and incorrect A_0 (see Sec. 3.2.1). The prompts are given in Fig. 7 and 8.

A.4 Distillation Details

After the data curation stage, training examples are categorized into two formats: (Q, S_0, C_1) for correct answers A_0 and (Q, S_0, C_1, T_1, S_1) for incorrect answers. For simplicity, instances with (Q, S_0, C_1) are padded to $(Q, S_0, C_1, T'_1, S'_1)$ using a predefined template (see Fig. 9). Here, T'_1 and S'_1 are negligible compared to the original reasoning T_0 and summary S_0 . During inference (as described in Alg. 1), if a critique C_n predicts correctness, T_n and S_n can be disregarded since the extracted answer A_n from S_n will match S_{n-1} .

B Detailed Experimental Setup

B.1 Training Data

This appendix outlines the filtering criteria applied to our candidate question pool, focusing on two primary principles: difficulty and answer verifiability. We detail the filtering pipeline for each dataset source as follows:

S1.1 Dataset The dataset comprises 1,000 high-quality questions together with their ground-truth answers sourced from multiple domains, with the reasoning trajectory annotated by DeepSeek-R1. However, some questions with open-ended answers can not be reliably evaluated using a rule-based

The Design of Critique Space

<critique>

Analysis:

The initial approach incorrectly assumes that "exactly two" and "exactly three" categories can be directly summed with those numbers to derive the count of residents owning all four items. This overlooks the distinction between "exactly two" (and "exactly three") and the inclusion of the universal set (candy hearts). The problem requires applying inclusion-exclusion principles to account for overlaps among the four items, not just summing those who own exactly two or three. The solution conflates the total population with direct arithmetic of exclusive categories, leading to an invalid conclusion.

Improvement suggestions:

1. Re-examine the definitions of "exactly two" and "exactly three" in the context of a universal set (candy hearts), recognizing that these groups inherently include ownership of the universal item.
2. Use inclusion-exclusion formulas to model overlaps between the four categories systematically.
3. Differentiate between residents owning all four items and those owning subsets (e.g., two specific items plus candy hearts).
4. Set up equations based on known totals (e.g., total residents, counts for each item) to solve for unknown overlaps.

Overall judgment:

Incorrect

</critique>

Figure 6: An Example of the Critique.

Prompt for Generating Critiques of Incorrect Answers

You are tasked with analyzing your last solution to a problem and providing constructive feedback based on previous solutions. Do NOT provide direct solutions.

You have already know your last solution to the problem is incorrect.

Important: Do NOT mention something like "you have already know your last solution is incorrect" in your feedback.

Structure your response using the following format (without <format> tags):

<format>

Analysis:

{ Analysis }

Improvement suggestions:

{ Suggestions }

Overall judgment:

{ Incorrect }

</format>

Figure 7: The Prompt for Generating Critiques of Incorrect Answers.

evaluation framework. After the removal of these unverifiable instances, we retain 861 questions with both validated answers and corresponding DeepSeek-R1 annotated reasoning trajectories. We will also use this subset as our \mathcal{D}_{direc} .

DeepMath-103K It contains 103K math problems, each annotated with two reasoning paths generated by DeepSeek-R1. We retain only 0.6K questions where the DeepSeek-R1-annotated reasoning paths are judged to be incorrect compared to ground truth answers by a rule-based evaluation method. We bypass the initial generation stage of Double-Checker and instead initialize the reasoning process directly from the DeepSeek-R1-annotated reasoning paths.

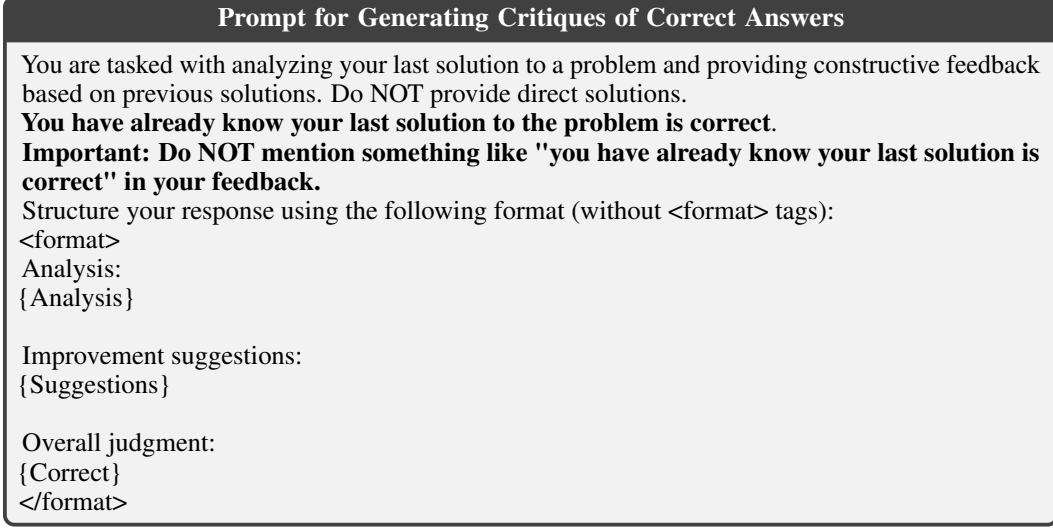


Figure 8: The Prompt for Generating Critiques of Correct Answers.

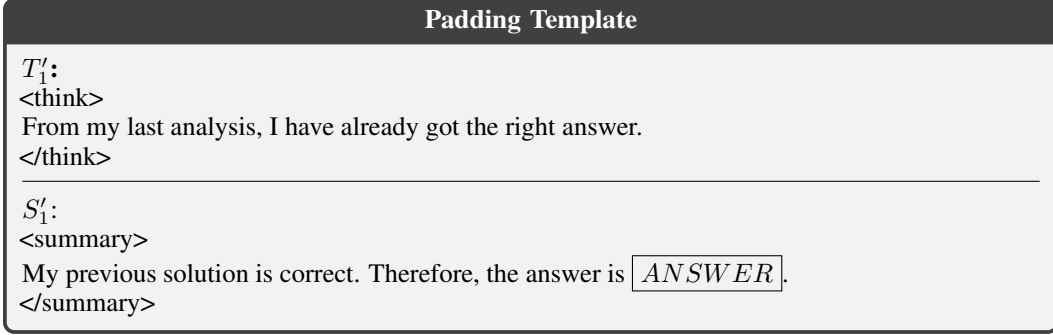


Figure 9: The Padding Template of Training Examples, where ANSWER is A_0 .

OpenRS & ORZ-Math-Hard The OpenRS dataset contains 7,000 mathematical reasoning problems, and ORZ-Math-Hard comprises 13,000 challenging math problems. Neither dataset provides DeepSeek-R1-annotated reasoning trajectories. To filter simple questions, we generate four responses per question using DeepSeek-R1-Distill-Qwen-7B with temperature 0.6 and retain only those questions where at least two responses are incorrect. For the remaining questions, we generate four responses using DeepSeek-R1-Distill-Qwen-32B with temperature 0.6 and retain only those with at least two incorrect responses. This yields 2.3K difficult problems from OpenRS and 4.4K from ORZ-Math-Hard.

To balance the distribution of correct and incorrect initial summaries (S_0), we also exclude correctly solved questions from DeepMath-103K, OpenRS, and ORZ-Math-Hard based on their initial generation by DeepSeek-R1.

B.2 Training Details

We train the DeepSeek-R1-Distill-Qwen-7B and DeepSeek-R1-Distill-Qwen-32B on our curated training set $\mathcal{D}_{\text{train}}$ using full-parameter fine-tuning. The 7B model is trained on either 8×H800 GPUs or 8×H20 GPUs, while the 32B model is trained on either 8×H800 GPUs or 32×H20 GPUs. The training process employs DeepSpeed ZeRO optimization [41] (stage 2 for 7B, and stage 3 for 32B) for efficient memory utilization and FlashAttention2 [42] for accelerated training. Following [10], we set the maximum sequence length to 16,384 tokens and use a batch size of 32, with a learning rate of 5×10^{-6} . Training times are approximately 0.7 hours per epoch for the 7B model on 8×H20 GPUs and 1.1 hours per epoch for the 32B model on 8×H800 GPUs.

B.3 Evaluation Details

We evaluate our models on six benchmarks covering diverse reasoning tasks. For mathematical reasoning, we use:

- AIME24/AIME25: Extremely difficult math competition problems, each dataset contains only 30 examples.
- MATH500 [48]: A high school level math problems, the subset of 500 problems is from the original test set of MATH [43].
- OlympiadBench [44]: A benchmark of Olympiad-level problems requiring advanced problem-solving skills. We only include the math subset for our evaluation.

For multidisciplinary reasoning, we use GPQA [12], which covers questions spanning biology, physics, and other domains.

Following [10], we adopt an unbiased pass@1 metric for datasets with limited test examples (AIME24 and AIME25), generating 16 samples with decoding temperature 0.6. For other benchmarks, we generate 4 samples per problem to report pass@1. For baseline models, we use the top-p parameter suggested in their original papers, while for our models, we fix top-p = 1.0. Inference is accelerated using vLLM [45], with a maximum sequence length of 32,768 tokens.

C Results Clarification

C.1 Results Source

As described in Section 4.2, most of the results for open-source LLMs presented in Table 1 are reproduced through our own experiments. However, for LLMs that require API access (e.g., OpenAI-o1-1217), we have cited the results from their official technical reports or other sources due to budget constraints. Similarly, the results for InfyThink [46] are sourced from their paper, as their code and models are not publicly accessible.

The sources of these results are detailed as follows:

- Results of InfyThink are from their paper [46].
- The results of OpenAI-o1-Preview on AIME24, MATH500 [43], AMC23, GPQA [12], and OlympiadBench-Math [44] are from LIMO [10].
- The results of OpenAI-o1-mini, OpenAI-o1-1217, DeepSeek-R1 on AIME24, MATH500, and GPQA are from DeepSeek-R1 report [15].
- The result of OpenAI-o1-mini on UGMATHBench is from UGMATHBench [4].
- The results of OpenAI-o1-Preview, OpenAI-o1-mini, and DeepSeek-R1 on AIME25 are from [49].

For the Self-Refine baseline, we follow a similar setting to the GSM8K dataset in (author?) [29], where the model is prompted with “There is an error in the solution above. To find the error, go through the semantically complete solution and check if everything looks good.” after the initial generation.

D More Results

D.1 About Generalizability

To evaluate the generalizability of our Double-Checker, we report additional results for the 7B model on LiveCodeBench [50] and the Business and Law subsets from MMLU-Pro [51]. As shown in Table 3, Double-Checker achieves performance improvements on LiveCodeBench, Business, and Law, despite not being trained on any coding-specific or social science examples. We believe that applying Double-Checker to training data from a broader and more diverse range of domains would further enhance its generalizability to previously unseen tasks. In particular, the data curation pipeline and training paradigm of Double-Checker are directly transferable to other domains.

Table 3: Performance comparison across different benchmarks.

Model	GPQA	LiveCodeBench	Business	Law
DeepSeek-R1-Distill-7B (Our initial model)	35.4	38.4	61.1	14.3
DeepSeek-R1-Distill-7B naive SFT	28.7	38.5	61.7	14.9
Double-Checker-7B	40.4	39.4	64.3	17.2

Table 4: Total number of input tokens (including problem and previous summary) per round.

	N=0	N=1	N=2	N=3
avg. input tokens	450	970	740	700
input structure	Q	$Q + S_0$	$Q + S_1$	$Q + S_2$

D.2 Additional Computational Costs Analysis

We provide additional details on computational costs in this appendix. Table 4 summarizes the total number of input tokens per round, including both the problem statement Q and the previous summary S_{N-1} . Our design discards intermediate reasoning content, retaining only summaries for subsequent rounds, which minimizes input size relative to output size. For $N = 0$, the input consists solely of Q , while for $N \geq 1$, the input comprises $Q + S_{N-1}$. As shown, input tokens are negligible compared to output tokens.

Additionally, Table 5 reports the average total inference time for Double-Checker on AIME24, using a single H20 GPU. The inference time per response for the “Naive SFT baseline” is 9.42 seconds, whereas Double-Checker achieves cumulative averages between 7.78 seconds ($N = 0$) and 9.47 seconds ($N = 1$). These results demonstrate that the added rounds do not significantly increase computational cost.

Table 5: The Average Inference time for Double-Checker on AIME24 using one H20 GPU.

	N=0	N=1	N=2	N=3
avg time for this round	7.78	1.69	1.49	0.95
avg cumulative time	7.78	9.47	10.96	11.92