TEMPORAL CONSISTENCY FOR LLM REASONING PROCESS ERROR IDENTIFICATION

Jiacheng Guo[1], Yue Wu[2], Jiahao Qiu[1], Kaixuan Huang[1], Xinzhe Juan[3], Ling Yang[1,2], and Mengdi Wang[1]

[1]Department of Electrical & Computer Engineering, Princeton University
[2]AI Lab, Princeton University
[3]Department of Computer Science & Engineering, University of Michigan

ABSTRACT

Verification is crucial for effective mathematical reasoning. We present a new *temporal consistency* method where verifiers iteratively refine their judgments based on the previous assessment. Unlike one-round verification or multi-model debate approaches, our method leverages consistency in a sequence of self-reflection actions to improve verification accuracy. Empirical evaluations across diverse mathematical process error identification benchmarks (Mathcheck, ProcessBench, and PRM800K) show consistent performance improvements over baseline methods. When applied to the recent DeepSeek R1 distilled models, our method demonstrates strong performance, enabling 7B/8B distilled models to outperform all 70B/72B models and GPT-4o on ProcessBench. Notably, the distilled 14B model with our method achieves performance comparable to Deepseek-R1. Our codes are available at https://github.com/jcguo123/Temporal-Consistency

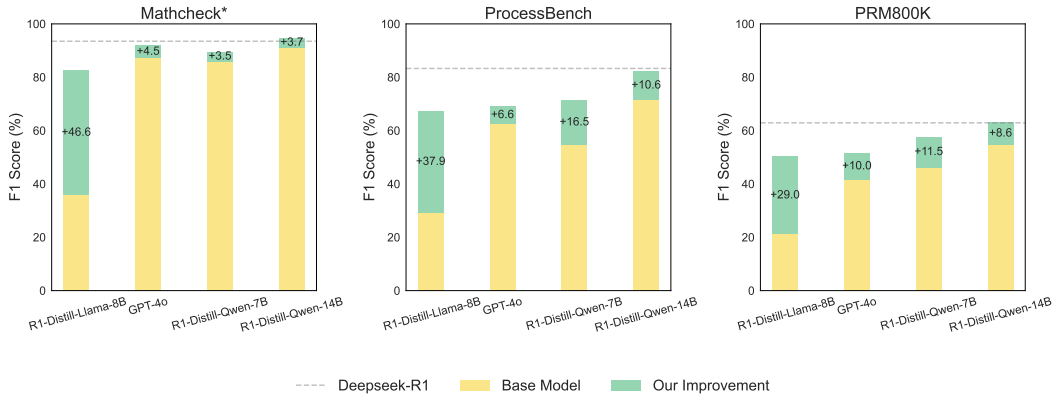

Figure 1: Performance improvements for various models on process error identification benchmarks.

1 INTRODUCTION

Large language models (LLMs) have shown impressive capabilities in reasoning tasks (Grattafiori et al., 2024; Yang et al., 2024b; Jaech et al., 2024; Guo et al., 2025; Yang et al., 2025), but still often make mistakes when generating complex multi-step solutions. To address this issue, Process Reward Models (PRMs) (Lightman et al., 2023; Luo et al., 2024b) have been introduced to guide generations. Instead of providing feedback solely on the final answer, PRMs evaluate every intermediate step in the reasoning chain, thereby aligning the model's chain of thought with correct logical sequences.

However, existing PRMs face several key limitations that hinder their broader applicability. First, training a PRM requires large-scale, high-quality annotated datasets, making the process highly data-intensive and costly to scale (Guo et al., 2025). Second, PRMs exhibit poor out-of-domain generalization; models trained on specific problem distributions often struggle to accurately evaluate

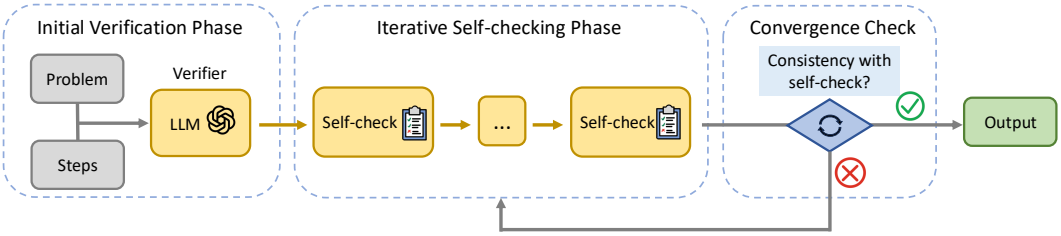

Figure 2: Overview of our Temporal Consistency approach, where each LLM iteratively examines its own verification results until reaching a stable result (stopping criteria defined in Section 2). The self-checking mechanism allows LLMs to refine their judgments based on previous verifications, potentially correcting initial misidentifications.

reasoning steps when confronted with diverse problem types (Zeng et al., 2025; Lin et al., 2024). Finally, the effectiveness of PRMs is intrinsically limited by the capability of the base model (Luo et al., 2024a). These challenges highlight the need for further research to develop more scalable process supervision techniques in LLMs.

An alternative way is to adopt some training-free approaches like majority voting (Wang et al., 2022) or debate-based approaches (Du et al., 2023), which have shown effectiveness in aggregating opinions and resolving conflicts between multiple reasoning trajectories.

Nevertheless, we found that both methods show limitations when applied to mathematical process error identification tasks. Majority voting often fails when errors are identified by only a minority of LLMs (Huang et al., 2024). Debate-based approaches sometimes struggle due to an asymmetry in mathematical reasoning: erroneous reasoning paths tend to generate lengthy, seemingly logical justifications, while correct reasoning paths provide only simple justifications. This asymmetry can cause debate methods to favor incorrect justifications, as more elaborate (though flawed) arguments may overshadow simple (but correct) justifications.

To address these limitations, we develop a simple but effective training-free approach to enhance process error identification capabilities. The intuition is to leverage the consistency between a sequence of self-reflection actions because the LLMs should be more likely to remain consistent and confident when asked to review correct validations. As shown in Figure 2, we propose the **Temporal Consistency** method, where each LLM iteratively checks its identifications, and the final output is only produced when multiple LLMs demonstrate consistent self-checking over time, effectively reducing unstable incorrect identifications.

We further evaluate our approach across three annotated mathematical step datasets, PRM800K (Lightman et al., 2023), ProcessBench (Zheng et al., 2024a), and MathCheck*[1] (Zhou et al., 2024). Our experiments demonstrated consistent performance gains across different models, benchmarks, and difficulty levels. We then conducted experiments on R1 distilled models (Guo et al., 2025), where our method achieved remarkable improvements: as shown in Figure 1 for Deepseek-R1-Distill-Llama-8B, improvements of **46.6%** on MathCheck*, **37.9%** on ProcessBench, and **29.0%** on PRM800K; for Deepseek-R1-Distill-Qwen-7B, improvements of **3.5%** on MathCheck*, **16.5%** on ProcessBench, and **11.5%** on PRM800K; for Deepseek-R1-Distill-Qwen-14B, improvements of **3.7%** on MathCheck*, **10.6%** on ProcessBench, and **8.6%** on PRM800K. Notably, our method enables distilled 7B/8B models to achieve **71.3%/67.2%** on ProcessBench, surpassing all existing 70B/72B models and GPT-4o reported in Zheng et al. (2024a). With our method applied, the distilled 14B model demonstrates performance comparable to Deepseek-R1's. As shown in Figure 3, our Temporal Consistency method establishes a new type of test-time scaling law. Unlike conventional approaches that scale by increasing the number of parallel samples, our method scales through iterative refinement over time (temporal dimension).

2 Methodology

In this section, we introduce our method that utilizes multiple rounds of validation to improve identification accuracy. We begin by defining the process error identification task.

[1]We use MathCheck* to denote a balanced dataset that combines MathCheck's process judging problems (containing only incorrect solutions) with problems with correct solutions from ProcessBench.

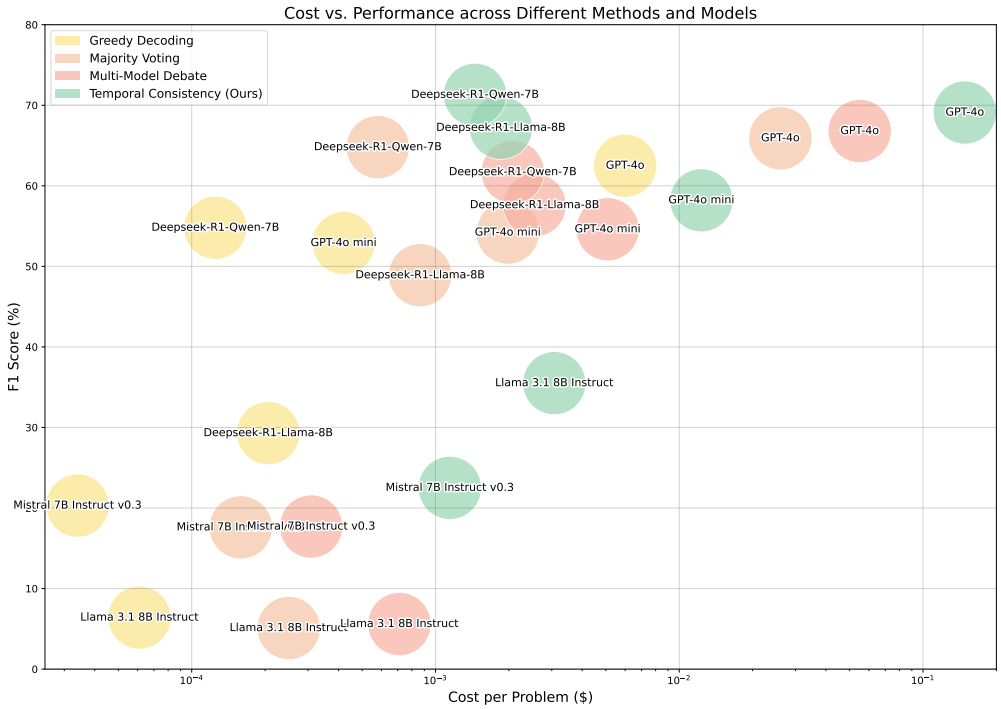

Figure 3: Cost v.s. Performance across different methods and models on ProcessBench. The x-axis (logarithmic scale) shows the cost per problem in dollars (based on OpenRouter pricing [2]), while the y-axis shows the F1 Score percentage.

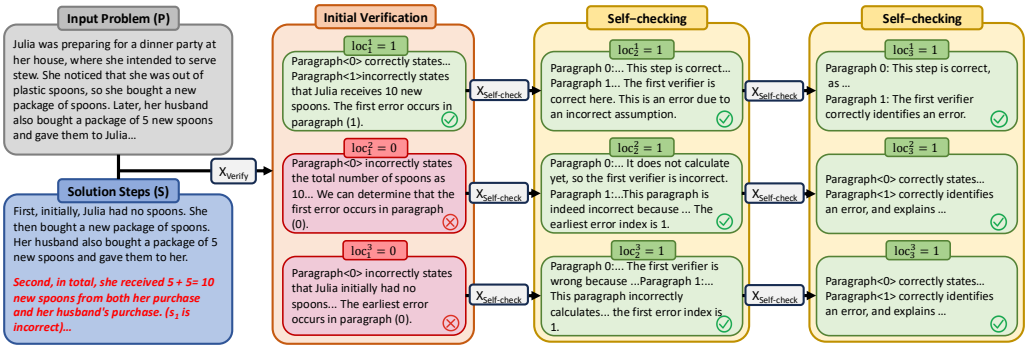

Figure 4: Example of the self-checking process: The first error occurred in step 1. Initially, two LLMs incorrectly identified the first incorrect step, while one correctly located the first incorrect step. After self-checking, all LLMs achieve the correct identification.

Task Definition Given a problem P and its step-by-step solution $S = \{s_0, s_1, \ldots, s_{n-1}\}$, where each s_i represents the i-th solution step, our objective is to identify the first incorrect step, if any, and output a location index $\texttt{loc} \in \{-1, 0, \ldots, n-1\}$. Here, $\texttt{loc} = -1$ indicating that all steps are correct, while for $\texttt{loc} \geq 0$, $s_{\texttt{loc}}$ represents the first incorrect step.

We now introduce the **Temporal Consistency** algorithm. This method adds a temporal dimension to the verification process by having each LLM consider its own previous assessment, leveraging consistency in a sequence of self-reflection. We employ K LLMs as verifiers, denoted by $\text{LLM}^1, \ldots, \text{LLM}^K$. The algorithm has three phases:

[2] https://openrouter.ai

Initial Verification Phase For each $i \in \{1, \ldots, K\}$, given a problem P, a solution S, and a designated process error identification prompt X_{Verify}, LLM^i examines the solution step by step. It identifies the location of the first incorrect step $\texttt{loc}_1^i$, and provides the corresponding reasoning response $\texttt{res}_1^i$:

$$(\texttt{loc}_1^i, \texttt{res}_1^i) = \text{LLM}^i(P, S, X_{\text{Verify}})$$

These initial verifications establish a set of independent assessments.

Iterative Self-checking Phase For time steps $t \geq 2$, let $(\texttt{loc}_{t-1}^i, \texttt{res}_{t-1}^i)$ represent the verification results from the previous iteration for each $i \in \{1, \ldots, K\}$. With a designated self-verification prompt $X_{\text{Self-check}}$, LLM^i performs a subsequent self-assessment:

$$(\texttt{loc}_t^i, \texttt{res}_t^i) = \text{LLM}^i(P, S, X_{\text{Self-check}}, \texttt{loc}_{t-1}^i, \texttt{res}_{t-1}^i).$$

The distinction between the initial verification phase and the self-checking phase is incorporating previous verification results to provide additional context. This temporal dependency enables the LLMs to potentially correct initial misidentifications. Figure 4 illustrates the self-checking mechanism.

Convergence Check After each iteration t, the algorithm determines the majority identification $\overline{\texttt{loc}}_t$ by applying a majority voting function:

$$\text{MajorityVote}(\texttt{loc}_t^1, \ldots, \texttt{loc}_t^K) = \underset{\texttt{loc} \in \{-1, \cdots, n-1\}}{\operatorname{argmax}} \left| \{i : \texttt{loc}_t^i = \texttt{loc}\} \right|. \tag{1}$$

This function aggregates the verification outcomes from K different LLMs and returns the error step that is most frequently identified. Specifically, $\left|\{i : \texttt{loc}^i = \texttt{loc}\}\right|$ counts the number of LLMs that have identified step $\texttt{loc}$ as incorrect. The algorithm then evaluates the stability of these identifications across all LLMs. Let p_t be the proportion of agents supporting $\overline{\texttt{loc}}_t$, formally defined as

$$p_t = \frac{\left|\{i : \texttt{loc}_t^i = \overline{\texttt{loc}}_t\}\right|}{K}. \tag{2}$$

When sufficient stability and consensus are reached, the algorithm terminates and outputs the final identification. Detailed stopping conditions defined with $\texttt{loc}_t$ and p_t are provided in Section 2.1.

This approach leverages the strengths of multiple independent verifications and consistency across the temporal dimension. By allowing each LLM to build on its previous assessments while remaining isolated from others, the algorithm minimizes the risk of reinforcing arguments that appear plausible but are incorrect. The complete algorithm is detailed in Algorithm 1.

2.1 Stopping Criteria

In practice, most agents converge to an identification within just a few rounds, making further self-checks computationally redundant. To enhance efficiency, we propose a heuristic stopping criterion that permits early termination for "high confidence" problems while allowing continued self-checking for "low confidence" problems.

For any round $t \in \{1, \ldots, T\}$, let $\overline{\texttt{loc}}_t$ denote the majority identification defined in equation 1, and p_t be the proportion of agents supporting $\overline{\texttt{loc}}_t$ defined in equation 2. Based on these definitions, we design two stopping conditions over q consecutive rounds, where q is a given consistency requirement:

1. **Majority Stability**:

$$\overline{\texttt{loc}}_{t-q+1} = \overline{\texttt{loc}}_{t-q+2} = \cdots = \overline{\texttt{loc}}_t,$$

2. **Growing Consensus**:

$$p_{t-q+1} \leq p_{t-q+2} \leq \cdots \leq p_t.$$

The majority stability condition requires that the majority identification remains unchanged over the past q rounds, ensuring a consistent outcome in majority voting. Concurrently, the growing consensus condition needs the proportion of agents supporting the majority identification to not decrease across these q rounds. The underlying intuition is that the correct answer should be identified with "increasing confidence" over the past q rounds.

The algorithm terminates when both conditions are satisfied or when the maximum number of rounds T is reached. The consistency requirement q is a parameter that can be adjusted according to task-specific requirements.

Algorithm 1 Temporal Consistency

Input: Problem P, solution S, number of LLMs K, initial verification prompt X_{Verify}, self-checking prompt $X_{\text{Self-check}}$, consistency requirement q, max rounds T.
/* Initial Verification Phase */
for $i = 1$ **to** K in parallel **do**
  $(\texttt{loc}_1^i, \texttt{res}_1^i) \leftarrow \text{LLM}^i(P, S, X_{\text{Verify}})$
end for
/* Iterative Self-checking Phase */
for round $t = 2$ **to** T **do**
  for LLM $i = 1$ **to** K in parallel **do**
    $(\texttt{loc}_t^i, \texttt{res}_t^i) \leftarrow \text{LLM}^i(P, S, X_{\text{Self-check}}, \texttt{loc}_{t-1}^i, \texttt{res}_{t-1}^i)$
  end for
  $\overline{\texttt{loc}}_t \leftarrow \text{MajorityVote}(\texttt{loc}_t^1, ..., \texttt{loc}_t^K)$
  $p_t \leftarrow |\{i : \texttt{loc}_t^i = \overline{\texttt{loc}}_t\}|/K$
  if $t \geq q$ **then**
    stable $\leftarrow \bigwedge_{j=0}^{q-2}(\overline{\texttt{loc}}_{t-j} = \overline{\texttt{loc}}_{t-q+1})$
    growing $\leftarrow \bigwedge_{j=0}^{q-2}(p_{t-j} \geq p_{t-j-1})$
    if stable **and** growing **then**
      return $\overline{\texttt{loc}}_t$
    end if
  end if
end for
return $\overline{\texttt{loc}}_T$ {Return final majority if max rounds reached}

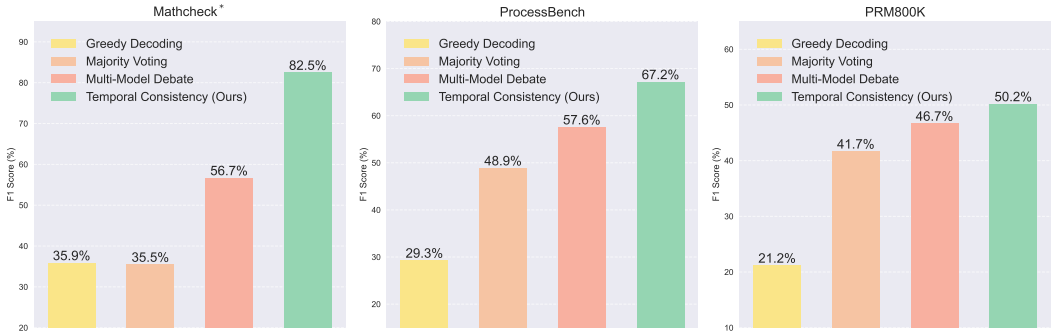

Figure 5: Performance comparison across three datasets (Mathcheck*, ProcessBench, and PRM800K). Our Temporal Consistency approach (green) consistently outperforms baseline methods, including greedy decoding (yellow), majority voting (orange), and multi-model debate (red).

2.2 Comparison with Existing Methods

Existing majority voting approaches (Cobbe et al., 2021; Li et al., 2022; Wang et al., 2022) perform multiple generations simultaneously, essentially scaling horizontally to enhance stability. In contrast, our method allows each LLM to build upon its previous assessments, achieving vertical scaling over time. This sequential self-reflection enables each verification to benefit from prior insights.

Moreover, our approach differs from multi-model debate methods (Du et al., 2023) in treating LLM independence. Although debate methods allow models to exchange information, thus enabling them to see other agents' answers and gain additional perspectives, this openness risks influence from persuasive yet incorrect arguments. For further illustration, an example can be found in Appendix C. In contrast, our method maintains strict isolation between LLMs. Each LLM focuses solely on its own reasoning process, thereby reducing the risk of propagating elaborate but erroneous arguments.

Model	Method	Mathcheck*	ProcessBench	PRM800K
GPT-4o mini	Greedy Decoding	78.8	52.9	34.0
	Majority Voting	80.4	54.2	37.9
	Multi-Model Debate	79.9	54.6	38.0
	Temporal Consistency (Ours)	**84.8**	**58.2**	**39.0**
GPT-4o	Greedy Decoding	87.3	62.5	41.6
	Majority Voting	89.0	65.9	42.6
	Multi-Model Debate	90.8	66.8	50.7
	Temporal Consistency (Ours)	**91.8**	**69.1**	**51.6**
Llama 3.1 8B Instruct	Greedy Decoding	13.3	6.4	2.4
	Majority Voting	5.9	5.1	6.8
	Multi-Model Debate	6.8	5.6	2.6
	Temporal Consistency (Ours)	**60.2**	**35.5**	**22.1**
Mistral 7B Instruct v0.3	Greedy Decoding	26.4	20.3	13.0
	Majority Voting	26.3	17.6	12.1
	Multi-Model Debate	26.2	17.7	12.1
	Temporal Consistency (Ours)	**37.4**	**22.5**	**13.3**

Table 1: Performance comparison across different models. Numbers represent F1 score (%). The best performance for each model is highlighted in bold. Our method consistently outperforms baselines across all models and benchmarks.

3 Experiments

3.1 Experimental Setup

Dataset We evaluate our method on ProcessBench (Zheng et al., 2024a), a comprehensive dataset combining multiple mathematical problem-solving benchmarks. The dataset consists of 3,400 problems from four sources: 400 from GSM8K (Cobbe et al., 2021), 1,000 from MATH dataset (Hendrycks et al., 2021), 1,000 from OlympiadBench (He et al., 2024), and 1,000 from Omni-MATH (Gao et al., 2024). Each problem includes both generated solutions and human-annotated processes. Additionally, we incorporate 516 process judging problems based on GSM8K from MathCheck (Zhou et al., 2024) and 300 randomly selected problems based on MATH dataset from PRM800K (Lightman et al., 2023). Since the process judging problem in MathCheck only contains incorrect solutions, we combine it with the GSM8K problems with correct steps from ProcessBench to create a balanced dataset, which we denote as MathCheck*. For PRM800K, we consider both 0 and 1 annotations as correct steps and -1 as incorrect steps. We evaluate the F1 score for all benchmarks, which is the harmonic mean of the accuracies on incorrect and correct samples.

Model	Method	Mathcheck*	ProcessBench	PRM800K
Deepseek-R1-Qwen-7B	Greedy Decoding	86.0	54.8	46.2
	Majority Voting	89.3	64.8	55.1
	Multi-Model Debate	84.8	61.7	51.2
	Temporal Consistency (Ours)	**89.5**	**71.3**	**57.7**
Deepseek-R1-Llama-8B	Greedy Decoding	35.9	29.3	21.2
	Majority Voting	35.5	48.9	41.7
	Multi-Model Debate	56.7	57.6	46.7
	Temporal Consistency (Ours)	**82.5**	**67.2**	**50.2**

Table 2: Performance comparison of Deepseek R1 distilled models on three benchmarks. Numbers represent F1 score (%). The best performance for each model is highlighted in bold.

Baseline Methods We compare our approach against three baseline methods: (1) Verification with greedy decoding (Zhang et al., 2022), where a single agent generates a verification deterministically, (2) Majority voting among multiple agents (Wang et al., 2022), where multiple agents independently generate verifications, and the final decision is made based on majority voting and (3) Verification with debate-based reasoning (Du et al., 2023), where multiple agents generate verifications independently, and they will receive the answer from the other agents and then generate a new identification.

Parameter Setting To ensure a fair comparison, we employ 5 parallel agents in each of the three methods: majority voting, debate-based verification, and our Temporal Consistency approach.

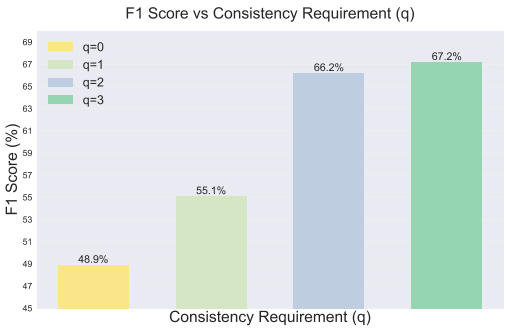

Figure 6: Performance comparison across different consistency requirements on ProcessBench for Deepseek-R1-Llama-8B. Higher consistency requirements, indicating stricter stability requirements, correlate with improved F1 scores.

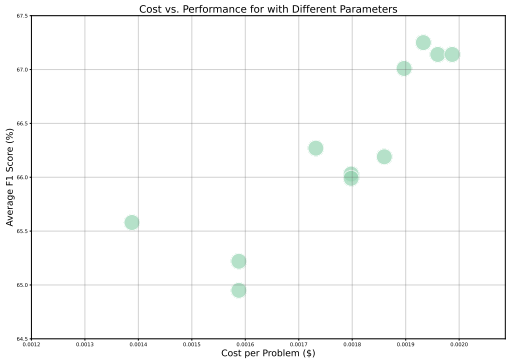

Figure 7: Cost-performance analysis of our method with different parameter configurations (max rounds and consistency requirement) on ProcessBench for Deepseek-R1-Llama-8B. The horizontal axis shows the cost per problem, while the vertical axis shows the average F1 score. As the computational budget increases, we observe improved performance.

Following Du et al. (2023), the debate method proceeds in two rounds: an initial verification round followed by a debate round. Our method implements convergence criteria requiring stability across 3 consecutive rounds, with a maximum of 10 rounds. We use Deepseek-R1-Llama-8B (Guo et al., 2025) in all our experiments except those in Table 1, Table 2 and Figure 1. Appendix B shows complete experimental configurations and implementation details.

3.2 Main Results

Improvement over Diverse Dataset Figure 5 presents the performance comparison across three datasets for Deepseek-R1-Llama-8B. Our Temporal Consistency approach consistently outperforms baseline methods across all evaluation settings.

On Mathcheck*, our method achieves an F1 score of 82.5%, showing an improvement of 46.6% over greedy decoding and 25.8% over multi-model debate. For ProcessBench, we observe consistent improvements with our method achieving 67.2% F1 score, compared to 29.3% for greedy decoding and 57.6% for multi-model debate. On PRM800K, our method maintains its advantage with 50.2% F1 score, showing a 29.0% improvement over greedy decoding.

Improvement over Different Base Models To demonstrate the generalizability of our approach, we conducted experiments across different language models, including GPT-4o mini, GPT-4o (Hurst et al., 2024), Llama 3.1 8B Instruct (Grattafiori et al., 2024) and Mistral 7B Instruct (Jiang et al., 2023). We evaluated these models on Mathcheck*, ProcessBench, and PRM800K. As shown in Table 1, our Temporal Consistency method consistently outperforms baseline methods across all benchmarks. This consistent performance across different models demonstrates the effectiveness of our approach.

Improvement for Distilled Models We further evaluate our method and the baseline methods on the recently released Deepseek R1 distilled models (Guo et al., 2025), including DeepSeek-R1-Distill-Qwen-7B and DeepSeek-R1-Distill-Llama-8B. As shown in Table 2, our Temporal Consistency method demonstrates remarkable effectiveness on 7B/8B-scale models, achieving 71.3% and 67.2% accuracy on ProcessBench with DeepSeek-R1-Distill-Qwen-7B and DeepSeek-R1-Distill-Llama-8B respectively, surpassing GPT-4o (69.1%) and all 70B/72B models reported in Zheng et al. (2024a), including Llama-3.3-70B-Instruct (58.0%), Qwen2.5-Math-72B-Instruct (45.5%) and Qwen2.5-72B-Instruct (61.2%) (Yang et al., 2024a).

3.3 Additional Analysis

Different Choice of Consistency Requirement We investigated the impact of different consistency requirements on model performance using ProcessBench. As shown in Figure 6, we experimented

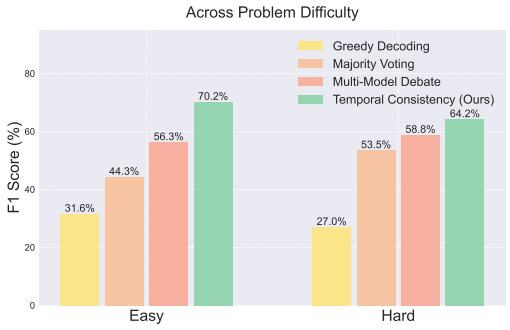

Figure 8: Performance comparison across problem difficulty levels. Problems are categorized as Easy (from GSM8K and MATH) or Hard (from OlympiadBench and Omni-MATH). Our method shows particular advantages on harder problems, maintaining more stable performance than baseline approaches.

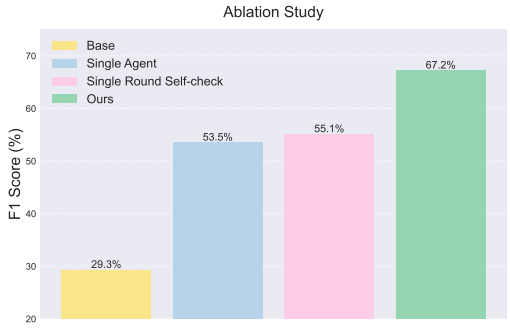

Figure 9: Ablation study results for ProcessBench demonstrating the effectiveness of both iterative generation and multi-agent components, with their combination yielding the best performance.

with consistency requirements ranging from 0 to 3, where higher values indicate stricter requirements for output stability. The F1 score demonstrates a consistent upward trend as the consistency requirement increases, starting from 48.9% without the self-checking requirement (parameter = 0) and reaching 67.2% with the strictest stability requirement (parameter = 3). This correlation suggests that requiring more stable outputs through multiple verification rounds leads to more accurate results.

Performance Across Problem Difficulty To analyze our method's effectiveness across varying complexity levels, we categorized ProcessBench problems into two groups following the difficulty definition in Zheng et al. (2024a): Easy (derived from GSM8K and MATH) and Hard (derived from OlympiadBench and Omni-MATH). Figure 8 illustrates the performance comparison across these categories. All methods demonstrate strong performance on easy problems, with our approach achieving 70.2% F1 score. The performance gap becomes more evident for hard problems, where our method maintains robust performance at 64.2%.

Cost-Performance Analysis To understand the trade-offs between computational resources and verification performance, we conducted experiments with various parameter configurations of our method. Figure 7 illustrates how performance scales with increased computational budget across different parameter settings. We observe a general trend where higher computational investment yields better verification results.

Ablation Study To understand the contribution of each component in our approach, we conducted an ablation study on ProcessBench, with results shown in Figure 9. We evaluated four configurations: the greedy decoding method, Temporal Consistency without multi-agent, self-checking without iterative generation, and our method. The results demonstrate that both components contribute to the overall performance. Starting from the base F1 score of 29.3%, each component independently improves performance, with the multi-agent self-checking and iterative mechanisms contributing improvements of 24.2% and 25.8%, respectively. The combination achieves the best performance with an F1 score of 67.2%.

4 Conclusion

We presented an Temporal Consistency approach for improving mathematical process error identification in language models. Our method leverages temporal consistency patterns in verification behavior, allowing LLMs to recheck their judgments through multiple rounds. We demonstrated how this approach effectively improves verification accuracy across different models and problem types through empirical evaluation.

Our key insight is that the temporal stability of verifications can serve as a reliable indicator of correctness. This finding opens new directions for developing methods focusing on consistency over time rather than agreement across agents. Our results suggest that incorporating temporal dynamics can enhance the reliability of mathematical reasoning methods.

REFERENCES

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17682–17690, 2024. 13

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. 5, 6, 13

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023. 2, 5, 6, 7, 13, 14

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D Goodman. Stream of search (sos): Learning to search in language. *arXiv preprint arXiv:2404.03683*, 2024. 13

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, et al. Omni-math: A universal olympiad level mathematic benchmark for large language models. *arXiv preprint arXiv:2410.07985*, 2024. 6, 13

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv e-prints*, pp. arXiv–2407, 2024. 1, 7

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*, 2025. 13

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. 1, 2, 7, 13

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. 6, 13

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. 6, 13

Zhenyu Hou, Xin Lv, Rui Lu, Jiajie Zhang, Yujiang Li, Zijun Yao, Juanzi Li, Jie Tang, and Yuxiao Dong. Advancing language model reasoning through reinforcement learning and inference scaling. *arXiv preprint arXiv:2501.11651*, 2025. 13

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. 13

Siyuan Huang, Zhiyuan Ma, Jintao Du, Changhua Meng, Weiqiang Wang, and Zhouhan Lin. Mirror-consistency: Harnessing inconsistency in majority voting. *arXiv preprint arXiv:2410.10857*, 2024. 2

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024. 7, 13

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. 1

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023. 7

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can llms actually correct their own mistakes? a critical survey of self-correction of llms. *Transactions of the Association for Computational Linguistics*, 12:1417–1440, 2024. 13

Jikun Kang, Xin Zhe Li, Xi Chen, Amirreza Kazemi, Qianyi Sun, Boxing Chen, Dong Li, Xu He, Quan He, Feng Wen, et al. Mindstar: Enhancing math reasoning in pre-trained llms at inference time. *arXiv preprint arXiv:2405.16265*, 2024. 13

Maxim Khanov, Jirayu Burapacheep, and Yixuan Li. Args: Alignment as reward-guided search. *arXiv preprint arXiv:2402.01694*, 2024. 13

Kuang-Huei Lee, Ian Fischer, Yueh-Hua Wu, Dave Marwood, Shumeet Baluja, Dale Schuurmans, and Xinyun Chen. Evolving deeper llm thinking. *arXiv preprint arXiv:2501.09891*, 2025. 13

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022. 5

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023. 1, 2, 6, 13

Yong Lin, Skyler Seto, Maartje ter Hoeve, Katherine Metcalf, Barry-John Theobald, Xuan Wang, Yizhe Zhang, Chen Huang, and Tong Zhang. On the limited generalization capability of the implicit reward model induced by direct preference optimization, 2024. URL https://arxiv.org/abs/2409.03650. 2

Runze Liu, Junqi Gao, Jian Zhao, Kaiyan Zhang, Xiu Li, Biqing Qi, Wanli Ouyang, and Bowen Zhou. Can 1b llm surpass 405b llm? rethinking compute-optimal test-time scaling. *arXiv preprint arXiv:2502.06703*, 2025. 13

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models by automated process supervision, 2024a. URL https://arxiv.org/abs/2406.06592. 2

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, et al. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024b. 1, 13

Ruilin Luo, Zhuofan Zheng, Yifan Wang, Yiyao Yu, Xinzhe Ni, Zicheng Lin, Jin Zeng, and Yujiu Yang. Ursa: Understanding and verifying chain-of-thought reasoning in multimodal mathematics. *arXiv preprint arXiv:2501.04686*, 2025. 13

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, 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. 13

Ning Miao, Yee Whye Teh, and Tom Rainforth. Selfcheck: Using llms to zero-shot check their own step-by-step reasoning, 2023. URL https://arxiv.org/abs/2308.00436. 13

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. 13

Jiahao Qiu, Yifu Lu, Yifan Zeng, Jiacheng Guo, Jiayi Geng, Huazheng Wang, Kaixuan Huang, Yue Wu, and Mengdi Wang. Treebon: Enhancing inference-time alignment with speculative tree-search and best-of-n sampling. *arXiv preprint arXiv:2410.16033*, 2024. 13

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024. 13

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314. 13

Vighnesh Subramaniam, Yilun Du, Joshua B Tenenbaum, Antonio Torralba, Shuang Li, and Igor Mordatch. Multiagent finetuning: Self improvement with diverse reasoning chains. *arXiv preprint arXiv:2501.05707*, 2025. 13

Zhengyang Tang, Ziniu Li, Zhenyang Xiao, Tian Ding, Ruoyu Sun, Benyou Wang, Dayiheng Liu, Fei Huang, Tianyu Liu, Bowen Yu, et al. Enabling scalable oversight via self-evolving critic. *arXiv preprint arXiv:2501.05727*, 2025. 13

Ziyu Wan, Xidong Feng, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. Alphazero-like tree-search can guide large language model decoding and training. In *Forty-first International Conference on Machine Learning*, 2024. 13

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)*, pp. 9426–9439, 2024. 13

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022. 2, 5, 6, 13

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*, 2024. 13

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024a. 7

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement, 2024b. URL https://arxiv.org/abs/2409.12122. 1

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E Gonzalez, and Bin Cui. Buffer of thoughts: Thought-augmented reasoning with large language models. *Advances in Neural Information Processing Systems*, 2024c. 13

Ling Yang, Zhaochen Yu, Tianjun Zhang, Minkai Xu, Joseph E Gonzalez, Bin Cui, and Shuicheng Yan. Supercorrect: Supervising and correcting language models with error-driven insights. *arXiv preprint arXiv:2410.09008*, 2024d. 13

Ling Yang, Zhaochen Yu, Bin Cui, and Mengdi Wang. Reasonflux: Hierarchical llm reasoning via scaling thought templates. *arXiv preprint arXiv:2502.06772*, 2025. 1, 13

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024. 13

Thomas Zeng, Shuibai Zhang, Shutong Wu, Christian Classen, Daewon Chae, Ethan Ewer, Minjae Lee, Heeju Kim, Wonjun Kang, Jackson Kunde, Ying Fan, Jungtaek Kim, Hyung Il Koo, Kannan Ramchandran, Dimitris Papailiopoulos, and Kangwook Lee. Versaprm: Multi-domain process reward model via synthetic reasoning data, 2025. URL https://arxiv.org/abs/2502.06737. 2

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*, 2024. 13

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*, 2022. 6, 13

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Processbench: Identifying process errors in mathematical reasoning, 2024a. URL https://arxiv.org/abs/2412.06559. 2, 6, 7, 8, 13, 14

Xin Zheng, Jie Lou, Boxi Cao, Xueru Wen, Yuqiu Ji, Hongyu Lin, Yaojie Lu, Xianpei Han, Debing Zhang, and Le Sun. Critic-cot: Boosting the reasoning abilities of large language model via chain-of-thoughts critic. *arXiv preprint arXiv:2408.16326*, 2024b. 13

Zihao Zhou, Shudong Liu, Maizhen Ning, Wei Liu, Jindong Wang, Derek F Wong, Xiaowei Huang, Qiufeng Wang, and Kaizhu Huang. Is your model really a good math reasoner? evaluating mathematical reasoning with checklist. *arXiv preprint arXiv:2407.08733*, 2024. 2, 6, 13

A Related Work

Datasets and Benchmarks for Process Error Detection Process error detection in mathematical reasoning requires annotations at the step level, currently available in three major datasets. PRM800K (Lightman et al., 2023) pioneered this direction by providing human-annotated reasoning steps based on the MATH dataset (Hendrycks et al., 2021), focusing on high school and college-level mathematics. MathCheck (Zhou et al., 2024) extends this approach to elementary mathematics by synthesizing solutions with incorrect steps from GSM8K problems (Cobbe et al., 2021), offering a systematic evaluation of step-by-step verification. Most recently, ProcessBench (Zheng et al., 2024a) expands the coverage of mathematical difficulty by providing expert-annotated solution steps across four distinct datasets: GSM8K, MATH, and notably, OlympiadBench (He et al., 2024) and Omni-MATH (Gao et al., 2024) for competition and olympiad-level challenges. Our experimental evaluation across these benchmarks provides comprehensive insights into our method's effectiveness from basic arithmetic to advanced mathematical reasoning.

Process Error Identification Methods Approaches to error detection in language models can be categorized into two main streams. The first focuses on training specialized verification models, such as process reward models (Lightman et al., 2023; Luo et al., 2024b; Setlur et al., 2024; Wang et al., 2024; Zhang et al., 2024) and finetuned language models (Cobbe et al., 2021; Kang et al., 2024; Zheng et al., 2024b; Yang et al., 2024d; Tang et al., 2025; Luo et al., 2025; Guan et al., 2025). While these training-based methods have shown promising results, they require additional training data and significant computational resources, especially for larger models. The second stream explores inference-time verification through prompting techniques like self-reflection (Miao et al., 2023; Madaan et al., 2024). Recent work has demonstrated that language models often struggle to correct errors without external feedback (Huang et al., 2023; Kamoi et al., 2024). Similar to self-reflection work (Madaan et al., 2024; Yang et al., 2024d), which iteratively generates improvement suggestions, our method employs an iterative process.

Rather than training new models, we focus on utilizing existing models more effectively. However, our Self-check method can also be applied to trained verification models to improve their accuracy potentially.

More General Reasoning Methods The broader field of reasoning in language models has explored various frameworks to enhance problem-solving capabilities and solution reliability. Chain-of-Thought prompting (Wang et al., 2022) and its variants like Tree-of-Thought (Yao et al., 2024) and Buffer-of-Thought (Yang et al., 2024c) have demonstrated that explicitly articulating intermediate reasoning steps improves model performance on complex reasoning tasks, and Zhang et al. (2024) further validates the effectiveness of reasoning in verification tasks. Predesigned reasoning structures (Zhang et al., 2022; Besta et al., 2024; Yang et al., 2024d) have also shown promise in improving mathematical capabilities by guiding LLMs to think along predefined trajectories. Multi-agent approaches such as debate mechanisms (Du et al., 2023; Subramaniam et al., 2025) enable models to critically examine solutions through structured discussions, while majority voting methods (Wang et al., 2022) generate multiple independent solutions and aggregate them through majority voting to enhance reliability. While each approach offers unique advantages, they demonstrate the importance of structured reasoning processes in improving model performance.

Test Time Scaling Recent studies have demonstrated that leveraging multiple samples during inference can significantly enhance model performance (Hurst et al., 2024; Guo et al., 2025; Yang et al., 2025). Through iterative refinement, models incorporate feedback from previous generations to guide subsequent outputs (Snell et al., 2024; Hou et al., 2025; Lee et al., 2025). While early approaches focused on simple majority voting strategies (Wang et al., 2022), subsequent research has advanced towards more sophisticated techniques, particularly in search-based methods (Khanov et al., 2024; Wan et al., 2024; Yang et al., 2025). The field has evolved with hybrid frameworks that seamlessly integrate tree-based search with sequential approaches (Wu et al., 2024; Snell et al., 2024; Qiu et al., 2024; Gandhi et al., 2024). Liu et al. (2025) conducted a study on optimizing test-time computation scaling across various policy models and problem complexities. Most closely related to our approach, Muennighoff et al. (2025) achieved substantial improvements in competition math questions by implementing parallel self-reflection on historical interactions.

B IMPLEMENTATION DETAILS

We use the gpt-4o-2024-08-06 API for GPT-4o and gpt-4o-mini API for GPT-4o-mini. We use Together API for the Deepseek-R1 model. All experiments can be performed on a single NVIDIA H100 GPU.

In the first round of all methods, the generation process was conducted using a temperature setting 0.7. The subsequent rounds vary slightly between closed-source and open-source models, with the following specifics:

- **Closed-source models:** For the debate method and our approach in later rounds, the temperature was set to 1.
- **Open-source models:** We used a fixed random seed of 42 throughout the experiments. For the debate method and subsequent rounds of our approach, the temperature was set to 0.7, and top-p=0.8, top-k=40.

B.1 PROMPTING STRATEGY FOR INITIAL VERIFICATION

In the first round of all methods, we utilized the verification prompts provided in Zheng et al. (2024a). The prompt format for the initial generation was:

```
The following is a math problem and a solution (split
into paragraphs, enclosed with tags, and indexed from
0):
[Math Problem]
{problem}
[Solution]
{tagged_response}
Your task is to review and critique the solution
paragraph by paragraph.  Once you identify an error in
a paragraph, return the index where the earliest error
occurs.  Otherwise, return the index of -1 (which
typically denotes "not found").
Please put your final answer (i.e., the index) in
\boxed{}.
```

B.2 DEBATE METHOD PROMPT ADAPTATION

The debate method is not designed for the verification task. To adapt it to our context, we combined the prompts for initial verification with those described in the appendix of Du et al. (2023). The adapted prompt is as follows:

```
These are the solutions to the problem from other
agents:
One agent solution:  {res['reason']}
Using the solutions from other agents as additional
information, please analyze this solution and end with
the earliest error index in \boxed{}, or -1 in \boxed{}
if no errors.
```

B.3 PROMPT FOR SECONDARY VERIFICATION

For the subsequent verification rounds in our approach, we designed a custom prompt to ensure thorough checking of the initial LLM's results. The prompt used is:

```
You are a secondary LLM for math problem solutions. The first
↪  verifier's task was to review math solutions paragraph by
↪  paragraph and identify the earliest error (if any), returning
↪ -1 if no errors were found.

The first verifier may have made mistakes. Your job is to
↪  carefully check their work. You will receive:

1. The original math problem

2. The solution steps

3. The first verifier's generated label (paragraph index where
↪  they found the first error, or -1)

4. The first verifier's reasoning

IMPORTANT:

1. You must check each paragraph carefully, as if you are
↪  re-solving the problem from scratch.

2. If you find any error|no matter how minor|locate the earliest
↪  paragraph containing that error.

3. If the solution is correct throughout, only then do you output
↪ -1.

4. The first verifier may be wrong. You cannot just accept their
↪  result. Always verify carefully and do not hesitate to
↪  disagree.

Your output format:

1. Begin with an overview: "Let's check the solution paragraph by
↪  paragraph based on the first agent's verification:"

2. For each paragraph (starting from paragraph 0), specify whether
↪  it is correct or not based on the first verifier's reason. If
↪  an error is found, explain the reason for the error. If
↪  correct, explain why it is correct.

3. End with the earliest error index in \boxed, or -1 in \boxed if
↪  no errors.
```

C EXAMPLES FOR VARIOUS METHODS

Example for Majority Voting (3 agents)

Problem: Julia was preparing for a dinner party at her house, where she intended to serve stew. She noticed that she was out of plastic spoons, so she bought a new package of spoons. Later, her husband also bought a package of 5 new spoons and gave them to Julia. While Julia was making the stew, she used three of the spoons to sample her stew. Later, when she went to set the table, she had a total of 12 spoons. How many spoons were in the package that Julia bought?

Solution Steps:

Step 0: Let's break down the problem step by step: First, initially, Julia had no spoons. She then bought a new package of spoons. Her husband also bought a package of 5 new spoons and gave them to her.
Step 1: Second, in total, she received 5 + 5 = 10 new spoons from both her purchase and her husband's purchase. First Error Step
Step 2: Third, while making the stew, she used 3 of the spoons to sample it, leaving her with 10 - 3 = 7 spoons.
Step 3: Fourth, when she went to set the table, she had a total of 12 spoons. We need to find out how many spoons were in the package that Julia bought initially.
Step 4: We know she received 10 more spoons after her husband gave her the additional 5 spoons. Since she had 7 spoons left after sampling and ended up with 12 spoons in total, the number of spoons she started with must have been 12 - 7 = 5.
Step 5: Therefore, the number of spoons in the package that Julia bought is 5.
Agent Analysis:

Agent 1:

- Paragraph ¡0¿ assumes Julia started with no spoons which isn't stated in problem
- This is an incorrect assumption as we don't know initial spoon count
- First error occurs in paragraph ¡0¿
- Therefore labels error as paragraph 0

Agent 2:

- Paragraph ¡0¿ contains unstated assumption about initial spoon count
- Making assumptions not given in original problem is an error
- First error identified in step ¡0¿
- Labels error as paragraph 0

Agent 3:

- Paragraph ¡0¿ correctly states initial conditions
- Paragraph ¡1¿ makes incorrect assumption about total spoons being 10
- First error found in calculating total spoons without knowing Julia's package
- Labels error as paragraph 1

Example for Change of Opinion After Debate (3 agents)

Problem: Amy is taking a history test. She correctly answers 80% of the multiple-choice questions, 90% of the true/false questions, and 60% of the long-answer questions. The multiple-choice and true/false questions are worth 1 point each, and the long answer questions are worth 5 points each. How many points does Amy score if there are 10 multiple-choice questions, 20 true/false questions, and 5 long answer questions?
Solution Steps:

Step 0: To calculate the total points that Amy scores on her history test, we need to determine how many points she earns from each type of question and then sum those points.
Step 1: First, for the multiple-choice questions: - There are 10 multiple-choice questions, each worth 1 point. - Amy answers 80% of them correctly. - Points from multiple-choice questions = 10 × 0.80 = 8 points.
Step 2: Second, for the true/false questions: - There are 20 true/false questions, each worth 1 point. - Amy answers 90% of them correctly. - Points from true/false questions = 20 × 0.90 = 18 points.
Step 3: Third, for the long-answer questions: - There are 5 long-answer questions, each worth 5 points. - Amy answers 60% of them correctly. - Points from long-answer questions = 5 × 0.60 × 5 = 15 points.
Step 4: Finally, to find the total points: - Total points = Points from multiple-choice + Points from true/false + Points from long-answer - Total points = 8 + 18 + 15 - Total points = 41
Step 5: Therefore, Amy scores a total of 41 points on her history test.
No Error ($\texttt{loc} = -1$)
Agent Analysis:

First Round:

- Agent 1: "After reviewing each paragraph step-by-step, no computational errors or mistakes in the logic of the solution are found. Therefore, index -1."
- Agent 2: "Upon reviewing each paragraph, no errors are found. While the calculation method in paragraph 3 is unconventional, it yields the correct result. Therefore, index -1."
- Agent 3: "Here, there is an error. The calculation should be 5 × 5 × 0.60 = 15 points, which is correct. However, the error is in the explanation that wrongly states the operation order as 5 × 0.60 × 5. While the final result is correct, the explanation could be misleading. The first minor error in explanation is found in Paragraph ¡3¿. Therefore, index 3."

Second Round (After Discussion):

- Agent 1: "Upon reviewing the solution and considering the feedback from other agents, I maintain that there are no computational errors affecting the final result. Index -1."
- Agent 2: "After reviewing the solutions from other agents, I now see that while the calculation gives the correct result, the explanation in paragraph 3 can be misleading. As Agent 3 pointed out, the operation order should be more clearly stated. Therefore, I revise my assessment to index 3."
- Agent 3: "Upon reviewing the solutions provided by other agents, I maintain my position. The calculation in paragraph 3, while giving the correct result, has a misleading explanation that could cause confusion. Index 3."

D Break Down of Evaluation Results

In this section, we provide the break down evaluation results in Table 1 and Table 2. Table 3 is the results for Mathcheck*, Table 4 are the results for PRM800K, Table 5 are the results for ProcessBench.

Table 3: Results for MathCheck*

Model Method	Err	Cor	F1
GPT-4o mini			
Greedy Decoding	75.0	82.9	78.8
Majority Voting	76.2	85.0	80.4
Multi-Model Debate	79.5	80.3	79.9
Temporal Consistency (Ours)	84.7	85.0	**84.8**
GPT-4o			
Greedy Decoding	84.5	90.2	87.3
Majority Voting	85.1	93.3	89.0
Multi-Model Debate	88.4	93.3	90.8
Temporal Consistency (Ours)	89.0	94.8	**91.8**
Llama 3.1 8B Instruct			
Greedy Decoding	44.6	7.8	13.3
Majority Voting	64.7	3.1	5.9
Multi-Model Debate	62.2	3.6	6.8
Temporal Consistency (Ours)	55.8	65.3	**60.2**
Mistral 7B Instruct v0.3			
Greedy Decoding	24.6	28.5	26.4
Majority Voting	15.9	76.2	26.3
Multi-Model Debate	15.7	79.3	26.2
Temporal Consistency (Ours)	34.1	41.5	**37.4**
Deepseek-R1-Llama-8B			
Greedy Decoding	67.6	24.4	35.9
Majority Voting	79.8	22.8	35.5
Multi-Model Debate	75.0	45.6	56.7
Temporal Consistency (Ours)	81.2	83.9	**82.5**
Deepseek-R1-Qwen-7B			
Greedy Decoding	77.9	95.9	86.0
Majority Voting	81.6	99.0	89.3
Multi-Model Debate	77.3	93.8	84.8
Temporal Consistency (Ours)	82.0	98.4	**89.5**

Table 4: Results for PRM800K

Model Method	Err	Cor	F1
GPT-4o mini			
Greedy Decoding	27.8	43.8	34.0
Majority Voting	31.3	47.9	37.9
Multi-Model Debate	34.4	42.5	38.0
Temporal Consistency (Ours)	34.4	45.2	**39.0**
GPT-4o			
Greedy Decoding	30.4	65.8	41.6
Majority Voting	30.4	71.2	42.6
Multi-Model Debate	41.9	64.4	50.7
Temporal Consistency (Ours)	39.2	75.3	**51.6**
Llama 3.1 8B Instruct			
Greedy Decoding	10.1	1.4	2.4
Majority Voting	18.9	4.1	6.8
Multi-Model Debate	23.3	1.4	2.6
Temporal Consistency (Ours)	15.0	42.5	**22.1**
Mistral 7B Instruct v0.3			
Greedy Decoding	11.5	15.1	13.0
Majority Voting	6.6	71.2	12.1
Multi-Model Debate	6.6	71.2	12.1
Temporal Consistency (Ours)	10.6	17.8	**13.3**
Deepseek-R1-Llama-8B			
Greedy Decoding	30.0	16.4	21.2
Majority Voting	41.0	42.5	41.7
Multi-Model Debate	42.3	52.1	46.7
Temporal Consistency (Ours)	39.2	69.9	**50.2**
Deepseek-R1-Qwen-7B			
Greedy Decoding	33.9	72.6	46.2
Majority Voting	41.9	80.8	55.1
Multi-Model Debate	38.8	75.3	51.2
Temporal Consistency (Ours)	44.5	82.2	**57.7**

Table 5: Results for ProcessBench

ProcessBench												
Model Method	GSM8K			MATH			OlympiadBench			Omni-MATH		
	Err	Cor	F1	Err	Cor	F1	Err	Cor	F1	Err	Cor	F1
GPT-4o mini Greedy Decoding	54.1	82.9	65.5	47.0	69.2	56.0	39.0	55.2	45.7	35.7	58.1	44.2
Majority Voting	56.0	85.0	67.5	47.8	71.6	57.3	38.9	60.5	47.3	36.1	58.1	44.5
Multi-Model Debate	63.8	80.3	71.1	52.9	64.4	58.1	42.1	49.9	45.6	40.3	47.7	43.7
Temporal Consi	63.3	85.0	**72.[illegible]**	51.3	74.1	**60.[illegible]**	43.1	60.8	**50.[illegible]**	41.2	61.0	**49.2**
GPT-4o Greedy Decoding	70.0	90.2	78.8	53.4	77.1	63.1	44.8	67.0	53.7	46.4	65.1	54.2
Majority Voting	73.4	93.3	82.2	53.9	82.5	65.2	48.3	72.8	58.0	49.2	71.4	58.3
Multi-Model Debate	77.8	93.3	**84.8**	61.4	77.0	68.4	53.7	59.5	56.4	56.1	58.9	57.5
Temporal Consi	74.9	94.8	83.[illegible]	58.1	90.1	**70.[illegible]**	45.8	86.7	**60.[illegible]**	48.7	86.3	**62.2**
Llama 3.1 8B Instruct Greedy Decoding	23.7	7.8	11.7	16.5	2.5	4.3	8.3	3.2	4.7	7.8	3.7	5.0
Majority Voting	41.1	3.1	5.8	30.6	1.7	3.3	19.8	4.1	6.8	25.4	2.5	4.5
Multi-Model Debate	45.9	3.6	6.7	37.9	3.7	6.7	30.6	2.9	5.4	32.0	2.5	4.6
Temporal Consi	34.8	65.3	**45.[illegible]**	28.8	51.5	**36.[illegible]**	23.8	37.5	**29.[illegible]**	24.6	40.7	**30.7**
Mistral 7B Instruct v0.3 Greedy Decoding	27.1	28.5	**27.8**	23.7	20.9	**22.2**	14.8	14.7	14.8	16.3	16.2	16.3
Majority Voting	12.6	76.2	21.6	11.8	69.7	20.2	7.6	65.8	13.6	8.4	67.2	15.0
Multi-Model Debate	12.6	79.3	21.7	12.0	70.2	20.4	7.3	67.0	13.1	8.7	66.0	15.4
Temporal Consi	20.8	41.5	27.[illegible]	19.4	25.9	22.[illegible]	18.0	19.8	**18.[illegible]**	16.2	31.5	**21.4**
Deepseek-R1-Llama-8B Greedy Decoding	44.9	24.4	31.6	45.5	24.1	31.5	35.1	24.8	29.0	31.2	20.7	24.9
Majority Voting	49.3	22.8	31.2	67.5	50.0	57.4	57.3	58.7	58.0	51.8	46.5	49.0
Multi-Model Debate	51.7	45.6	48.5	64.5	63.8	64.1	56.1	71.1	62.7	49.9	61.0	54.9
Temporal Consi	56.5	83.9	**67.[illegible]**	67.0	79.6	**72.[illegible]**	57.0	78.5	**66.[illegible]**	53.1	75.1	**62.2**
Deepseek-R1-Qwen-7B Greedy Decoding	52.2	95.9	67.6	50.5	80.0	61.9	39.0	64.6	48.7	29.6	66.0	40.9
Majority Voting	57.5	99.0	72.7	64.3	88.4	74.5	48.1	81.7	60.6	39.0	75.5	51.4
Multi-Model Debate	58.0	93.8	71.7	59.8	84.7	70.1	45.8	71.1	55.7	37.7	71.4	49.3
Temporal Consi	62.8	98.4	**76.[illegible]**	69.5	94.3	**80.[illegible]**	54.5	90.6	**68.[illegible]**	46.1	86.7	**60.2**