A Theoretical Study on Bridging Internal Probability and Self-Consistency for LLM Reasoning

Zhi Zhou [1] **Yuhao Tan** [1] **Zenan Li** [2] **Yuan Yao** [1]
Lan-Zhe Guo [1,3] **Yu-Feng Li** [1,4*] **Xiaoxing Ma** [1*]
[1]State Key Laboratory of Novel Software Technology, Nanjing University, China
[2]Department of Computer Science, ETH Zurich, Switzerland
[3]School of Intelligence Science and Technology, Nanjing University, China
[4]School of Artifical Intelligence, Nanjing University, China
zhouz@lamda.nju.edu.cn, liyf@nju.edu.cn, xxm@nju.edu.cn

Abstract

Test-time scaling seeks to improve the reasoning performance of large language models (LLMs) by adding computational resources. A prevalent approach within the field is *sampling-based test-time scaling methods*, which enhance reasoning by generating multiple reasoning paths for a given input during inference. However, despite its practical success, the theoretical foundations remain underexplored. In this paper, we provide the first theoretical framework for analyzing sampling-based test-time scaling methods, grounded in the perspective of confidence estimation. Based on the framework, we analyze two dominant paradigms: self-consistency and perplexity, and reveal key limitations: self-consistency suffers from high estimation error while perplexity exhibits substantial modeling error and possible degradation of the estimation error convergence. To address these limitations, we introduce RPC, a hybrid method that leverages our theoretical insights through two key components: *Perplexity Consistency* and *Reasoning Pruning*. *Perplexity Consistency* combines the strengths of self-consistency and perplexity, boosting the convergence rate of estimation error from linear to exponential while preserving model error. *Reasoning Pruning* prevents degradation by eliminating low-probability reasoning paths. Both theoretical analysis and empirical results across seven benchmark datasets demonstrate that RPC has a strong potential for reducing reasoning error. Notably, RPC achieves reasoning performance comparable to self-consistency while not only enhancing confidence reliability but also reducing sampling costs by 50%. The code and resources are available at https://wnjxyk.github.io/RPC.

1 Introduction

Recent advances in large language models (LLMs) have demonstrated their remarkable reasoning capabilities across diverse applications, including problem-solving [33, 35], planning [53, 13], and decision-making [45, 48]. Test-time scaling methods [58, 57, 20] can further enhance reasoning performance with additional computation. Among these strategies, the sampling-based test-time scaling method has emerged as a simple yet effective technique. Through generating multiple reasoning paths and selecting the most plausible one through a confidence estimation mechanism, such as self-consistency [57] and perplexity [10], this approach has achieved significant improvements.

The success of sampling-based test-time scaling hinges on accurately estimating the confidence of sampled reasoning paths, facilitating their comparison and enabling the selection of the most

*Corresponding author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

plausible answer in the Best-of-N manner [20, 29]. Current confidence estimation approaches can be categorized into two main types: (i) *consistency-based methods*, which use pre-defined consistency functions [11] to determine answer confidence, with self-consistency as the primary representative; and (ii) *probability-based methods*, which utilize scores from either the internal LLM probability [44] or external model [29] to evaluate the confidence of reasoning paths, with perplexity as the leading example. While both have proven effective in practice, a rigorous theoretical understanding remains lacking regarding their underlying mechanisms, inherent limitations, and potential improvements.

This paper addresses this gap by introducing the first theoretical framework for sampling-based test-time scaling methods in LLM reasoning. Within this framework, reasoning error is decomposed into two components: *Estimation Error* and *Model Error*. We apply our framework to analyze two representative methods from each category, self-consistency and perplexity, and identify their limitations. Self-consistency, which relies on Monte Carlo estimation, achieves only a linear convergence rate in estimation error, resulting in unsatisfactory performance when the sampling budget is limited. Perplexity utilizes the internal probabilities of LLMs to achieve exponential convergence rates, but they suffer from high model error, which compromises their effectiveness. Moreover, its convergence rate depends on the magnitude of probabilities and deteriorates significantly when probabilities are low. These insights provide guidance for developing improved methods: *An optimal approach should simultaneously achieve rapid estimation error convergence while maintaining low model error*.

To address the above limitations, we introduce a novel ***R**easoning-pruning **P**erplexity **C**onsistency* (RPC) method, which consists of two components: *Perplexity Consistency* and *Reasoning Pruning*. *Perplexity Consistency* integrates the internal probability of the LLM into the self-consistency framework, enabling rapid estimation error reduction that transitions from linear to exponential convergence while maintaining low model error. *Reasoning Pruning* mitigates the degradation issue of estimation error reduction rate when the internal LLM probability magnitude is low by automatically modeling the probability distribution and removing low-probability reasoning paths. Our theoretical analysis shows that RPC achieves both fast estimation error convergence and low model error, offering significant potential to reduce reasoning error. Our empirical results demonstrate the effectiveness and efficiency of RPC on seven benchmark datasets. Specifically, on four mathematical reasoning datasets, RPC reduces the required sampling budget by at least 50% while maintaining the same level of reasoning performance as self-consistency. When using the same sampling budget, RPC achieves an average improvement of 1.29% over existing methods. Additionally, RPC provides confidence estimates that are more closely aligned with the ground truth compared to existing methods.

To summarize, the main contributions of the paper are:

(1) We propose a theoretical framework that formulates sampling-based test-time scaling in LLM reasoning and decomposes reasoning error into estimation error and model error. This framework enables an analysis of existing techniques, offering guidance for developing improved approaches.

(2) Based on this framework, we introduce a novel test-time scaling method, RPC, which leverages the internal probability of LLMs within the self-consistency paradigm and removes low-probability reasoning paths, thereby accelerating error reduction and improving reasoning performance.

(3) Our theoretical analysis demonstrates that RPC achieves rapid estimation error convergence, providing strong potential for reducing reasoning error. Empirical results on seven benchmark datasets show that RPC substantially reduces the required sampling budget, improves reasoning accuracy, and enhances the reliability of confidence estimation.

2 Problem and Analysis

In this section, we first formulate the LLM reasoning problem using sampling-based test-time scaling methods. Then, we theoretically decompose the LLM reasoning error into *Estimation Error* and *Model Error*, and analyze two representative methods of consistency-based and probability-based methods. Our analysis reveals insights for building an advanced LLM reasoning method.

2.1 Problem Formulation

Given a reasoning problem (x, y), where x represents the input query, and y represents the ground-truth answer. The LLM generates a reasoning path $t = (t_1, \ldots, t_m)$ by sequentially sampling tokens

according to the conditional probability distribution $p(t_i \mid x, t_{<i})$, where m denotes the length of the reasoning path. The probability of generating the reasoning path $\hat{t}$ is defined as $p(\hat{t} \mid x)$, a.k.a the confidence of the reasoning path $\hat{t}$. An extraction function $g(\cdot)$ maps the reasoning path to the final answer $\hat{y} = g(\hat{t})$. We can further extend the probability to the answer $\hat{y}$, i.e., the answer confidence, denoted as $p(\hat{y} \mid x)$. Take the mathematical reasoning problem as an example, the reasoning path $\hat{t}$ could be "1 + 1 = 2. The answer is 2." and the function $g(\cdot)$ extracts the answer from $\hat{t}$, resulting in $\hat{y} = g(\hat{t}) = 2$. The reasoning correctness is evaluated by the indicator function $\mathbb{I}[\hat{y} = y]$.

The confidence represents the probability that the reasoning path $\hat{t}$ or answer $\hat{y}$ is correct, allowing LLMs to select the most reliable solution from multiple candidates in the Best-of-N manner [20, 29]. However, in practice, accessing the confidence for all possible reasoning paths or answers of LLMs is computationally infeasible. Therefore, we typically estimate the confidence by sampling finite n reasoning paths $\tilde{t}_1, \ldots, \tilde{t}_n$ from the LLM sampling distribution $p(t \mid x)$, which yields the estimated confidence $\hat{p}(\hat{t} \mid x)$ or $\hat{p}(\hat{y} \mid x)$ for any reasoning path $\hat{t}$ or answer $\hat{y}$.

To measure the reasoning performance of LLMs, we use the squared error [22] to penalize reasoning error of confidence for any single reasoning path $\hat{t}$ or answer $\hat{y}$:

$$\mathcal{E}_{\hat{p}}(\hat{t}) = \mathbb{E}\left[\left(\hat{p}(\hat{t} \mid x) - \mathbb{I}[g(\hat{t}) = y]\right)^2\right], \quad \mathcal{E}_{\hat{p}}(\hat{y}) = \mathbb{E}\left[\left(\hat{p}(\hat{y} \mid x) - \mathbb{I}[\hat{y} = y]\right)^2\right]. \tag{1}$$

where the expectation is taken over all possible combinations of n sampled reasoning paths $\tilde{t}_1, \ldots, \tilde{t}_n$, which are used to estimate the confidence $\hat{p}$. For notational clarity and simplicity, we omit the explicit form of expectation here and throughout the remainder of the paper.

In the following analysis, we categorize sampling-based test-time scaling methods into two types based on confidence estimation aspects: consistency-based methods and probability-based methods. Consistency-based methods [63, 72] estimate confidence by evaluating the agreement among different reasoning paths, with self-consistency [57] serving as a representative approach. Probability-based methods [20, 10] estimate confidence using either the internal probability provided by the LLM or the scores from external models, with perplexity [57] being a representative example. Our further analysis is conducted on each representative method.

2.2 Theoretical Analysis

In this section, we theoretically decompose the reasoning error into estimation error and model error. Next, we examine the specific forms of reasoning error for two representative methods, self-consistency (SC) and perplexity (PPL), to provide insights for improved algorithm design.

2.2.1 Reasoning Error Decomposition

Take the reasoning error $\mathcal{E}_{\hat{p}}(\hat{y})$ of $\hat{p}(\hat{y} \mid x)$ in Equation 1 as an example, we can decompose the reasoning error into the estimation error and model error as follows.

Proposition 1 (Error Decomposition)**.** *For any input x with ground-truth answer y and any possible answer $\hat{y}$, let $\hat{p}(\hat{y} \mid x)$ denote the unbiased estimated confidence of $\hat{y}$ and $p(\hat{y} \mid x)$ denote the ground truth confidence. Then, the reasoning error $\mathcal{E}_{\hat{p}}(\hat{y})$ can be divided into two components:*

$$\mathcal{E}_{\hat{p}}(\hat{y}) = \underbrace{\mathbb{E}\left[\left(\hat{p}(\hat{y} \mid x) - p(\hat{y} \mid x)\right)^2\right]}_{\textit{Estimation Error}} + \underbrace{\left(p(\hat{y} \mid x) - \mathbb{I}[\hat{y} = y]\right)^2}_{\textit{Model Error}}, \tag{2}$$

where the expectation is taken over sampled reasoning paths $\tilde{t}_1, \ldots, \tilde{t}_n$ for estimating confidence.

Remark 1. The detailed proof is provided in Appendix A.1. Proposition 1 separates the effect of confidence estimation on reasoning error from the effect of the LLM's reasoning capability. The *Estimation Error* depends solely on the sampling size and the confidence estimation strategy, while the *Model Error* is invariant and determined by the LLM's reasoning capability. This proposition demonstrates that, apart from the fixed *Model Error*, which is determined by the LLM's inherent reasoning capability, the reasoning error is bounded by the *Estimation Error*. Moreover, this proposition provides insights into two directions for improving LLM reasoning performance: (1) reducing the *Estimation Error* through larger sampling sizes or more accurate confidence estimation methods, and (2) reducing the *Model Error* by enhancing the LLM's reasoning capabilities or developing more sophisticated and effective confidence metrics.

Next, we analyze two representative methods in sampling-based test-time scaling: self-consistency (SC) from consistency-based methods and perplexity (PPL) from probability-based methods, using our theoretical framework. Below, we adopt a common assumption that LLM sampling follows a Bernoulli distribution [55], allowing us to compute the estimation error for specific methods.

2.2.2 Analysis of Self-Consistency

Self-consistency [60, 1, 4] is the representative method for consistency-based methods, which samples n reasoning paths $\tilde{t}_1, \ldots, \tilde{t}_n$, and estimates the confidence of any $\hat{y}$ using Monte-Carlo estimation by

$$\hat{p}^{(\mathrm{SC})}(\hat{y} \mid x) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}[\tilde{y}_i = \hat{y}], \quad \tilde{y}_i = g(\tilde{t}_i). \tag{3}$$

Then, the reasoning error of SC for a given problem (x, y) and any possible $\hat{y}$ can be computed by

$$\mathcal{E}_{\hat{p}^{(\mathrm{SC})}}(\hat{y}) = \mathbb{E}\left[\Big(\frac{1}{n} \sum_{i=1}^{n} \mathbb{I}[\tilde{y}_i = \hat{y}] - \mathbb{I}[\hat{y} = y] \Big)^2 \right]. \tag{4}$$

Finally, we conduct decomposition on SC to illustrate the key factors affecting the reasoning error.

Proposition 2 (SC Reasoning Error Decomposition)**.** *For any input x with ground-truth answer y, let $\hat{p}^{(\mathrm{SC})}(\hat{y} \mid x)$ denote the estimated probability for any possible answer $\hat{y}$ by* SC*. Then, the reasoning error $\mathcal{E}_{\hat{p}^{(\mathrm{SC})}}(\hat{y})$ can be divided into two components:*

$$\mathcal{E}_{\hat{p}^{(\mathrm{SC})}}(\hat{y}) = \underbrace{\frac{1}{n} p(\hat{y} \mid x)(1 - p(\hat{y} \mid x))}_{\textit{Estimation Error}} + \underbrace{\big(p(\hat{y} \mid x) - \mathbb{I}[\hat{y} = y]\big)^2}_{\textit{Model Error}}. \tag{5}$$

Remark 2. The detailed proof is provided in Appendix A.1. This proposition reveals that the estimation error of SC consists solely of variance since the sampling is unbiased. It decreases only linearly with increasing sample size, which results in substantial reasoning error when sampling is limited. This analysis suggests that a promising direction for improving SC is to develop methods that achieve faster estimation error convergence rates.

2.2.3 Analysis of Perplexity

Perplexity is a representative method for probability-based methods that directly utilizes the internal LLM probability $p(\hat{t} \mid x)$ for any reasoning path $\hat{t}$. However, since the number of possible reasoning paths is nearly infinite, the ground-truth probability can only be accessed for those paths that are actually sampled. Therefore, the estimated probability of any possible reasoning path $\hat{t}$ is estimated using the unique set of n sampled reasoning paths $\mathcal{R} = \mathrm{Set}\left(\tilde{t}_1, \ldots, \tilde{t}_n\right)$.

$$\hat{p}^{(\mathrm{PPL})}(\hat{t} \mid x) = \begin{cases} p(\tilde{t}_i \mid x), & \text{if} \quad \hat{t} = \tilde{t}_i, \\ 0, & \text{otherwise.} \end{cases} = \sum_{\tilde{t} \in \mathcal{R}} \mathbb{I}\left[\hat{t} = \tilde{t}\right] p(\tilde{t} \mid x). \tag{6}$$

Similarly, the reasoning error of PPL is denoted as follows, with the following proposition decomposing the reasoning error of PPL.

$$\mathcal{E}_{\hat{p}^{(\mathrm{PPL})}}(\hat{t}) = \mathbb{E}\left[\Big(\sum_{\tilde{t} \in \mathcal{R}} \mathbb{I}\left[\hat{t} = \tilde{t}\right] p(\tilde{t} \mid x) - \mathbb{I}[g(\hat{t}) = y] \Big)^2 \right]. \tag{7}$$

Proposition 3 (PPL Reasoning Error Decomposition)**.** *For any given input x with ground-truth answer y, let $\hat{p}^{(\mathrm{PPL})}(\hat{t} \mid x)$ denote the estimated probability of $\hat{t}$ by* PPL *method for any possible reasoning path $\hat{t}$. Then, the reasoning error $\mathcal{E}_{\hat{p}^{(\mathrm{PPL})}}(\hat{t})$ can be divided into two components:*

$$\mathcal{E}_{\hat{p}^{(\mathrm{PPL})}}(\hat{t}) = \underbrace{(1 - p(\hat{t} \mid x))^n p(\hat{t} \mid x)(2\mathbb{I}[\hat{y}_i = y] - p(\hat{t} \mid x))}_{\textit{Estimation Error}} + \underbrace{\big(p(\hat{t} \mid x) - \mathbb{I}[g(\hat{t}) = y]\big)^2}_{\textit{Model Error}}. \tag{8}$$

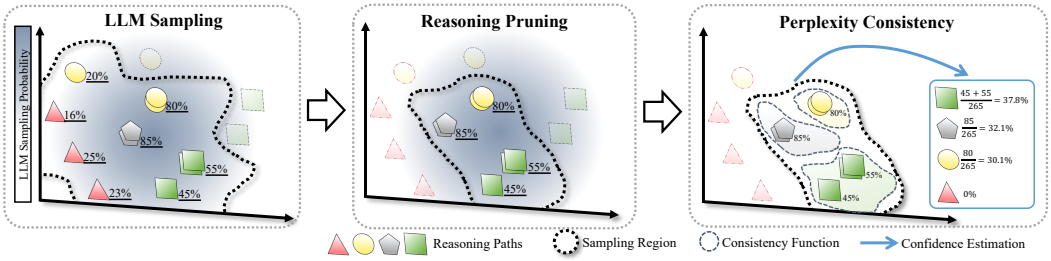


Figure 1: Illustration of the RPC approach. The *Reasoning Pruning* filters out low-probability answers, while the *Perplexity Consistency* incorporates LLM probabilities into the self-consistency framework, resulting in faster convergence of estimation error.

Remark 3. The detailed proof is provided in Appendix A.1. Compared with SC, the estimation error of PPL decreases exponentially, with the rate depending on the magnitude of ground-truth confidence $p(\hat{t} \mid x)$. For reasoning paths that are likely to yield the correct answer (i.e., those with non-negligible confidence), PPL achieves a substantially faster convergence rate. However, for reasoning paths with extremely low confidence, the convergence advantage of PPL degrades. Furthermore, the *Model Error* of PPL is typically larger than that of SC in practice, and we formally demonstrate this in the ideal case in Appendix 4. This analysis suggests that the degradation issue of *Estimation Error* and the large *Model Error* issue are fundamental challenges for improving PPL method.

3 Methodology

Based on our theoretical analysis, we identify three fundamental challenges in existing methods: (1) SC suffers from slow *Estimation Error* convergence, leading to efficiency concerns; (2) PPL exhibits large *Model Error*, compromising its effectiveness; (3) PPL's *Estimation Error* convergence advantage deteriorates significantly for certain cases, resulting in a degradation issue.

To address these challenges, we propose *Reasoning-Pruning Perplexity Consistency* (RPC), a novel method with two key components. First, we integrate internal LLM probabilities into the self-consistency framework to create *Perplexity Consistency* (PC), a confidence estimation function that reduces estimation error efficiently while maintaining low model error, thus addressing the first two challenges with theoretical guarantees. Second, we introduce a *Reasoning Pruning* (RP) module that resolves the third challenge by systematically filtering out reasoning paths with low probabilities.

3.1 Perplexity Consistency

To address the efficiency and effectiveness challenges simultaneously, we propose PC, which directly leverages the LLM's prediction probability like PPL, obtaining the benefit of an exponential convergence rate, and also applies the consistency function of SC to minimize the model error. Formally, for the unique set of n sampled reasoning paths $\mathcal{R} = \text{Set}\left(\tilde{t}_1, \ldots, \tilde{t}_n\right)$, the estimated probability of any possible answer $\hat{y}$ is

$$\hat{p}^{(\text{PC})}(\hat{y} \mid x) = \sum_{\tilde{t} \in \mathcal{R}} \mathbb{I}[g(\tilde{t}) = \hat{y}] p(\tilde{t} \mid x), \tag{9}$$

which calculates the cumulative probability of all unique reasoning paths $\left\{\tilde{t}_i \mid \tilde{t}_i \in \mathcal{R} \text{ and } g(\tilde{t}_i) = \hat{y}\right\}$ whose answer is $\hat{y}$. Therefore, the squared error of PC for any possible answer $\hat{y}$ is

$$\mathcal{E}_{\hat{p}^{(\text{PC})}}(\hat{y}) = \mathbb{E}\left[(\hat{p}^{(\text{PC})}(\hat{y} \mid x) - \mathbb{I}[\hat{y} = y])^2\right]. \tag{10}$$

Now, we present the following theorem, which explores the reasoning error decomposition of PC.

Theorem 4 (PC Reasoning Error Decomposition)**.** *Assume that* $k = |\{\tilde{t} \mid g(\tilde{t}) = \hat{y}\}|$ *and define* $\alpha := 1 - \frac{1}{k} p(\hat{y} \mid x)$*. Then, the reasoning error* $\mathcal{E}(\hat{p}^{(\text{PC})})$ *of* PC *can be divided into two components:*

$$\mathcal{E}_{\hat{p}^{(\text{PC})}}(\hat{y}) = \underbrace{\alpha^n p(\hat{y} \mid x)\left(2\mathbb{I}[\hat{y} = y] - (1 + \alpha^n) p(\hat{y} \mid x)\right)}_{\textit{Estimation Error}} + \underbrace{\left(p(\hat{y} \mid x) - \mathbb{I}[\hat{y} = y]\right)^2}_{\textit{Model Error}}.$$

Remark 5. The proof is presented in Appendix A.3. The theorem states that PC successfully fuses the strengths of PPL and SC: it achieves the same level of model error as SC while ensuring the similar convergence rate as PPL in the estimation error. Particularly, the convergence rate can be computed as $\alpha^n p(\hat{y} \mid x) = (1 - \frac{1}{k} p(\hat{y} \mid x))^n p(\hat{y} \mid x)$, which is exponential.

Remark 6. The convergence rate is primarily influenced by the magnitude of $p(\hat{y} \mid x)$. In most scenarios, it remains exponential, facilitating rapid estimation error reduction. However, when $p(\hat{y} \mid x) \to 0$ and $np(\hat{y} \mid x) \ll 1$, we only have $\alpha^n \to \frac{1}{1+np(\hat{y} \mid x)}$ [32], resulting in the convergence rate unexpectedly degenerating to a linear result.

3.2 Reasoning Pruning

To address the third degradation challenge, which is also verified in Remark 6, we propose directly pruning these sampled answers, called *Reasoning Pruning* (RP). The main idea of RP is that an answer with negligible $p(\hat{y} \mid x)$ is not likely to be the correct answer, so we may prune it directly without a significant increase in reasoning error. Specifically,RP sets $\hat{p}(\hat{y} \mid x) = 0$ when the cumulative probability of all its corresponding reasoning paths is lower than a threshold τ, making the estimation error vanish. However, there are two questions raised for RP: (1) Although pruning can boost the estimation error reduction, does it potentially increase the model error and thereby increase the overall reasoning error? (2) How should we determine the threshold τ for pruning?

For the first question, we theoretically prove that RP can achieve the optimal error reduction for each given problem (x, y) with the optimal threshold τ at a high probability.

Theorem 7 (Effectiveness of Reasoning Path Pruning)**.** *Assume that the optimal threshold $\tau = p(y \mid x)$, and let $\hat{k} = |\{\tilde{t}_i \mid g(\tilde{t}_i) = \hat{y}, i = 1, \ldots, n\}|$, which refers to the size of samples whose answer is $\hat{y}$. Hence,* RP *achieves the optimal error reduction with at least the probability*

$$1 - \exp\Big(- 2\hat{k}k^2(1 - \frac{\tau}{1-\alpha})^2\Big).$$

Remark 8. The proof is included in Appendix A.4. The theorem provides a guarantee that RP can achieve the optimal error reduction for each given problem (x, y) at a high probability. Note that the optimal error reduction not only boosts the estimation error reduction efficiency but also effectively reduces the model error, thus improving the final reasoning capability of LLMs.

For the second question, we develop an automated strategy to determine the τ based on the distribution of all sampled reasoning paths. Inspired by open-set recognition [5], we model the probability distribution of Ω_1 and Ω_2 as a mixture of two Weibull distributions, representing high and low probability regions. Elaborately, we define the PDF of the mixture distribution as:

$$f(x) = w_1 f_{\mathrm{W}}(x; k_1, \lambda_1) + w_2 f_{\mathrm{W}}(x; k_2, \lambda_2),$$

where the Weibull PDF [59] is defined as $f_{\mathrm{W}}(x; k, \lambda) = \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} \exp\left(-(\frac{x}{\lambda})^k\right)$. We use the maximum likelihood estimation methods to estimate the parameters, i.e., (k_1, λ_1), (k_2, λ_2), w_1, and w_2 on the probability distribution of all sampled reasoning paths for each reasoning problem. We assume that Weibull(k_1, λ_1) is the high probability distribution and Weibull(k_2, λ_2) is the other. Then, the probability of the reasoning path $\hat{t}$ being in the high probability distribution is derived by

$$P_{\text{High}}(x) = \frac{w_1 f_{\mathrm{W}}(x; k_1, \lambda_1)}{w_1 f_{\mathrm{W}}(x; k_1, \lambda_1) + w_2 f_{\mathrm{W}}(x; k_2, \lambda_2)}.$$

where x is the value of LLM internal probability. Then, we remove sampled reasoning paths $\tilde{t}$ satisfying $P_{\text{High}}(\hat{p}(\tilde{t} \mid x)) < 0.5$, which are more likely to be in the low probability distribution. Moreover, to ensure the algorithm's stability when n is limited, we employ the Truncated Mean method [43], retaining outputs with a probability greater than the overall mean. This prevents the removal of too many reasoning paths due to the potential inaccurate estimation of the mixture distribution.

3.3 RPC Method

Overall, we apply the *Reasoning Pruning* to all sampled reasoning paths $\tilde{t}_1, \ldots, \tilde{t}_n$ for removing low probability reasoning paths and then compute the confidence based on *Perplexity Consistency*, forming our proposed **R**easoning-pruning **P**erplexity **C**onsistency (RPC) confidence estimation method. The pseudo-code is shown in Algorithm 1 in Appendix B. Figure 1 illustrates its complete framework.

Table 1: Efficiency comparison of *Perplexity Consistency* module (PC) and RPC. The table shows the minimum number of samples needed to exceed the best performance of SC, with reduction rates in bold when sampling is reduced.

Method	MATH		MathOdyssey		OlympiadBench		AIME	
	Accuracy	#Samplings	Accuracy	#Samplings	Accuracy	#Samplings	Accuracy	#Samplings
Best of SC	50.57	64	28.32	112	11.07	128	9.40	128
PC	50.63	32	28.51	112	11.07	128	9.00	64
Δ	+0.06	**-50.0%**	+0.19	-0.0%	0.00	-0.0%	0.00	**-50.0%**
RPC	51.16	32	29.31	32	11.07	64	9.50	48
Δ	+0.59	**-50.0%**	+0.99	**-71.4%**	0.00	**-50.0%**	+0.10	**-62.5%**

4 Experiments

In this section, we conduct experiments to answer the following research questions:

RQ1: Efficiency. How does RPC reduce the number of samples required to achieve comparable performance through faster convergence?
RQ2: Efficacy. How does RPC improve reasoning performance compared to existing methods?
RQ3: Reliability. How does RPC enhance the reliability of confidence estimation compared to existing methods?

Moreover, further discussions are devoted to further demonstrating the effectiveness of RPC.

4.1 Experimental Setting

In this section, we briefly introduce the comparison methods, datasets, and implementation details. Due to space limitations, detailed experimental settings are included in Appendix C.

Comparison Methods. We compare three types of LLM confidences: perplexity confidence [57] (PPL), self-consistency confidence [10] (SC), and verbalized confidence [52] (VERB). The verbalized confidence is computed based on the probability that the LLM outputs "True" versus "False" when asked an "Is-True" question. For code generation tasks, we extracted verbalized confidence scores from the model's numerical likelihood expressions by prompting the LLM.

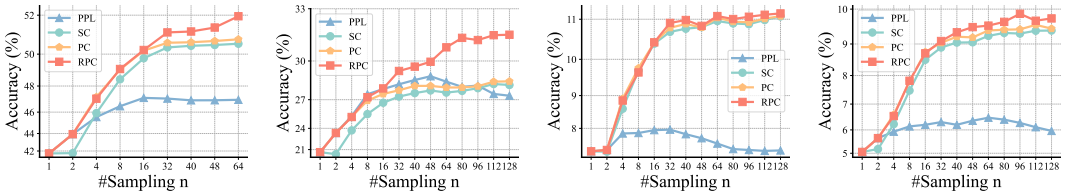
Datasets. We introduce four popular benchmarks for math reasoning: MATH [25], MathOdyssey [16], OlympiadBench [23], and AIME [67] (contains problems from 1983 to 2024). As to code generation tasks, we evaluate each method on three benchmarks, i.e., HumanEval [8], MBPP [3], and introductory-level problems of APPS [24].

Implementation Details. For math reasoning tasks, we evaluate the InternLM2-Math-Plus models with 1.8B and 7B parameters [65], as well as the DeepSeekMath-RL 7B model [49]. The consistency function $\mathbb{I}_C$ is the answer comparison. For code generation tasks, we evaluate the Deepseek-Coder 33B model. The consistency function $\mathbb{I}_C$ is constructed based on semantic equivalence [42] by clustering code based on given test cases. We set the sample size to $n = 128$ for the MathOdyssey, OlympiadBench, and AIME datasets and $n = 64$ for the MATH dataset by default. Each experiment is repeated 10 times with different random seeds, and the average performance is reported. All experiments were conducted on Linux servers with A800 and H800 GPUs.

4.2 Empirical Results

RQ1: Efficiency. How does RPC reduce the number of samples required to achieve comparable performance through faster convergence?

We evaluate our proposed RPC against the standard self-consistency method using four mathematical benchmark datasets with the InternLM-2-MATH-Plus 7B model. For the MATH dataset, we set the reasoning path size to 64, while we set the number of reasoning paths to 128 for the other datasets with SC. We then record the best performance and minimum sampling requirements for SC. For both



(a) MATH (b) MathOdyssey (c) OlympiadBench (d) AIME

Figure 2: The accuracy of the InternLM-2-MATH-Plus 7B model on four math reasoning datasets with different sample sizes n. Our proposed RPC achieves the best performance across all datasets, validating our theoretical analysis. We also show the performance of a single perplexity consistency module (denoted as PC), which further supports our theoretical findings.

Table 2: Performance Comparison using InternLM-2-MATH-Plus 7B model measured by accuracy and expected calibration error metrics. The best performance is highlighted in **bold**. The results show that our RPC outperforms existing methods in majority of cases.

Method	MATH		MathOdyssey		OlympiadBench		AIME		Average	
	Accuracy(↑)	ECE(↓)	Accuracy(↑)	ECE(↓)	Accuracy(↑)	ECE(↓)	Accuracy(↑)	ECE(↓)	Acc.(↑)	ECE(↓)
PPL	46.99 ± 0.20	48.99 ± 0.19	27.35 ± 1.22	67.70 ± 1.22	7.27 ± 0.36	86.90 ± 0.35	5.96 ± 0.48	88.98 ± 0.49	21.90	73.14
VERB	26.14 ± 0.25	47.46 ± 0.07	10.06 ± 0.61	69.92 ± 0.88	3.68 ± 0.16	84.68 ± 0.25	3.17 ± 0.17	86.29 ± 0.20	10.76	72.09
SC	50.57 ± 0.17	6.71 ± 0.18	28.25 ± 0.60	12.23 ± 0.54	11.07 ± 0.15	20.20 ± 0.16	9.40 ± 0.21	14.35 ± 0.23	24.82	13.37
RPC	**51.95 ± 0.15**	**6.41 ± 0.18**	**31.62 ± 0.75**	**9.87 ± 0.73**	**11.14 ± 0.15**	**18.86 ± 0.18**	**9.74 ± 0.23**	**14.32 ± 0.21**	**26.11**	**12.37**

RPC and our *Perplexity Consistency* module (denoted as PC), we report the minimum number of samples needed to match or exceed the performance of the SC in Table 1.

The results of PC indicate improved convergence rates compared to SC in several cases, while maintaining similar rates in others. These findings support the rapid convergence and degeneration issues of PC in Theorem 4. RPC shows significant efficiency improvements by requiring fewer samples to achieve comparable performance relative to SC. Moreover, the degeneration issues observed in PC are effectively addressed in RPC, validating both the effectiveness of the *Reasoning Pruning* module and our Theorem 7.

RQ2: Efficacy. How does RPC improve reasoning performance compared to existing methods?

We evaluate the performance of PC and RPC in Figure 2 across various sample budgets. The results demonstrate that RPC achieves better performance than both PPL (which relies on internal LLM probabilities) and SC (which employs Monte Carlo sampling). The detailed accuracy results, including mean and standard deviation in Table 2 support these findings.

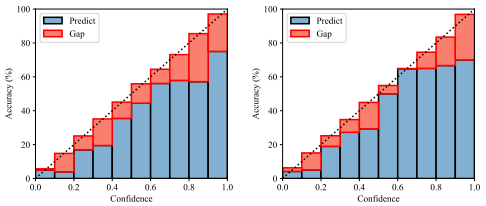
We also analyze the performance of PC separately. The results indicate that PC has a faster convergence rate than SC, which aligns with Theorem 4. The significant performance gains from PC to RPC, as shown in Figure 2a and Figure 2b, validate the effectiveness of the *Reasoning Pruning* module. This suggests that *Reasoning Pruning* helps reduce model errors when the LLM exhibits good alignment by eliminating incorrect reasoning paths that carry low LLM probability scores.

RQ3: Reliability. How does RPC enhance the reliability of confidence estimation compared to existing methods?

To evaluate the reliability of confidence estimation, we analyze the ECE results of RPC and comparison methods in Table 2. ECE measures the difference between predicted probabilities and empirical accuracy, as directly computing estimation error using ground-truth probabilities is virtually impractical. The results demonstrate that our RPC approach achieves lower ECE values and higher accuracy compared to PPL and SC, indicating more reliable confidence estimation. We visualize this improvement through reliability diagrams comparing SC and RPC in Figure 3 on MathOdyssey, which clearly shows the reduced calibration error of RPC.

4.3 Further Discussion

Performance on Code Generation Tasks. To investigate whether our proposed approaches can generalize to other reasoning tasks, such as code generation tasks, we evaluate RPC and comparison



(a) SC (b) RPC

Figure 3: The reliability diagrams of SC and RPC on MathOdyssey dataset using InternLM-2-MATH-Plus 7B model.

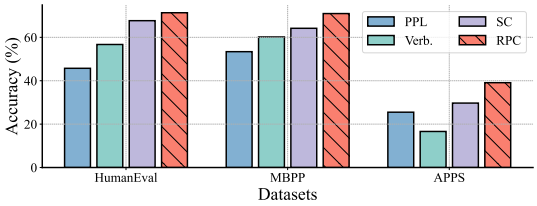


Figure 4: Performance on three code generation tasks using Deepseek-Coder 33B model. The experimental results show that our RPC achieves the best performance.

methods on three code generation benchmarks, as illustrated in Figure 4. The results show that RPC achieves the highest accuracy across all datasets, demonstrating its effectiveness beyond mathematics.

Performance on Additional Reasoning Tasks. To further validate the effectiveness of RPC, we conducted experiments on the GPQA [47] and LogiQA [39] benchmarks using the DeepSeek-R1-Distill-Qwen-7B model [21] with 16 samples. The results in Appendix D.5 are consistent with those from both math reasoning and code generation tasks, where RPC outperforms existing methods.

Performance across Model Scales and Architectures. To evaluate the generalization ability of our approaches across different model scales and architectures, we conducted additional experiments using InternLM2-Math-Plus 1.8B and DeepSeek-Math 7B models. The results in Table 3 is consistent with results in Table 2. We additionally report the detailed performance of InternLM2-Math-Plus 1.8B using diverse sampling budgets in Appendix D.2. The experimental results demonstrate the effectiveness of RPC across different model scales and architectures.

Performance using Advanced Methods and Models. To further verify the effectiveness of RPC across diverse scenarios, we conduct additional experiments using the DeepSeek-R1-Distill-Qwen-7B model [21], which is an advanced model with thinking capability. The results in Appendix D.4 demonstrate that the performance gains of RPC persist when combined with advanced models that have strong reasoning performance, highlighting its potential for enhancing LLM reasoning capability. Moreover, we also combine RPC with two advanced methods: ESC [36] and BoN using the reward model [69], which are advanced versions of SC and PPL, respectively. The results in Appendix D.4 show that RPC applied to both methods consistently outperforms the SC and their original versions, demonstrating its strong compatibility for integration with more advanced methods.

Performance using High Sampling Temperatures. As discussed above, sampling more diverse reasoning paths is important for improving reasoning performance. Therefore, we additionally conduct experiments under high sampling temperatures. As shown in Appendix D.1, RPC can further improve reasoning performance under high sampling temperatures by benefiting from the diverse reasoning paths, while the SC may deteriorate due to increased estimation errors at high temperatures.

Discussion about Computational Overhead. We discuss the computational overhead of RPC in Appendix D.7. Theoretically, RPC introduces only minimal computational overhead compared to SC. In practice, the additional computational overhead of RPC is negligible compared to the primary computational bottleneck (i.e., LLM inference time) in reasoning tasks. RPC actually provides an excellent computational trade-off, where the minimal computational overhead is exchanged for significant time savings achieved by reducing the number of required LLM inferences.

Discussion about Hyper-parameters. We provide a detailed discussion of the hyperparameters in Appendix D.6. Overall, RPC introduces the RP module to automatically prune reasoning paths with low probability without requiring manual threshold setting. This makes RPC a hyperparameter-free method that is robust across diverse reasoning tasks.

5 Related Work

This paper is related to the two topics, i.e., LLM Reasoning Boosting and LLM Confidence Estimation.

LLM Reasoning Boosting. Recent research [62, 38] has developed various methods to enhance LLM reasoning. CoT [31] proposes the "Let's think step by step" prompt to guide LLMs in generating

Table 3: Performance Comparison of different models and different parameter scales. The accuracy is reported as the mean and stdev. The best performance is highlighted in **bold**. The results show that our RPC outperforms existing methods in major cases.

Method	InternLM2-Math-Plus 1.8B				DeepSeekMath-RL 7B			
	MATH	MathOdyssey	OlympiadBench	AIME	MATH	MathOdyssey	OlympiadBench	AIME
PPL	33.24 ± 0.24	$\mathbf{16.56 \pm 0.88}$	3.08 ± 0.20	1.66 ± 0.15	42.51 ± 0.23	22.34 ± 1.00	5.90 ± 0.31	3.37 ± 0.46
VERB	7.21 ± 0.17	2.81 ± 0.26	0.77 ± 0.06	0.26 ± 0.05	14.29 ± 0.23	2.55 ± 0.24	2.36 ± 0.15	1.91 ± 0.12
SC	36.48 ± 0.15	14.52 ± 0.46	5.99 ± 0.17	2.66 ± 0.20	53.33 ± 0.09	36.68 ± 0.65	11.29 ± 0.17	9.42 ± 0.23
RPC	$\mathbf{37.88 \pm 0.16}$	16.35 ± 0.61	$\mathbf{6.52 \pm 0.24}$	$\mathbf{3.26 \pm 0.24}$	$\mathbf{53.37 \pm 0.11}$	$\mathbf{37.25 \pm 0.69}$	$\mathbf{11.30 \pm 0.11}$	$\mathbf{9.52 \pm 0.31}$

structured solutions. For complex problems, Least-to-Most [72] introduces a decomposition strategy that breaks down challenges into manageable sub-problems. Few-shot methods [58, 17, 70] leverage carefully selected examples to improve reasoning performance. To enable more comprehensive reasoning, search-based methods [19] integrate Monte Carlo Tree Search (MCTS). Recent advancements have further enhanced MCTS by incorporating reward models [68, 46]. To directly optimize reasoning abilities, researchers have explored fine-tuning approaches [66, 34, 38, 65] using specialized datasets and reinforcement learning techniques [49, 41]. While these methods focus on generating accurate reasoning paths, our RPC can build upon them by utilizing multiple sampling strategies, enabling better reasoning performance. Recent studies recognize its computational issues and propose early stopping [37] and dynamic sampling [56, 54, 2] to improve efficiency. While previous studies primarily focus on developing specific methods that demonstrate empirical effectiveness, our paper introduces a theoretical framework for analyzing them and provides theoretical insights.

LLM Confidence Estimation. The confidence estimation for LLM can be categorized into three types: (1) perplexity confidence, (2) verbalized confidence, and (3) self-consistency confidence. Perplexity confidence [28, 15] utilizes the geometric mean of LLM prediction probabilities (i.e., perplexity [10, 6]) to evaluate model adherence [44] and prompt quality [64]. Verbalized confidence [30, 60, 52] directly asks LLMs to express their confidence through various approaches, such as multi-agent deliberation [61], multi-step evaluation [60], top-k ranking [52], few-shot prompting [40], and reflection [14, 71]. Self-consistency confidence [57, 11, 12, 26, 50] measures the agreement among multiple generated answers to improve reasoning performance, with recent work [60, 1, 4] further developing this approach as a confidence metric. Recent studies have identified that LLM confidence is crucial for enhancing LLM reasoning, such as DeepConf [18], CISC [51], and TTSC [27], which align with our core idea. Our work provides theoretical explanations to support their findings.

6 Conclusion

In this paper, we present a theoretical framework for LLM reasoning problem in sampling-based test-time scaling, providing insights and principled guidance for designing better LLM reasoning methods. This framework decomposes LLM reasoning error into estimation error and model error, revealing the efficiency issues of the self-consistency method and effectiveness and degeneration issues of the perplexity method. To this end, we introduce ***R**easoning-pruning **P**erplexity **C**onsistency* (RPC), which integrates internal LLM probabilities into the self-consistency framework for faster estimation error convergence while maintaining low model error and prunes low-probability reasoning paths to address the degeneration issue. Both theoretical analysis and empirical results demonstrate that RPC achieves superior error convergence, reasoning performance, and confidence reliability.

Limitations and Future Work. Despite our theoretical analysis providing insights and guidance for designing better LLM reasoning methods, this paper presents only a preliminary step toward building advanced reasoning approaches. We believe our theoretical framework can guide future research in this promising direction. Detailed limitations and future works are discussed in Appendix E.

Broader Impacts

This paper presents a new theoretical framework for analyzing sampling-based test-time scaling methods in LLM reasoning, enabling researchers to identify the limitations of existing methods and to provide insights for developing future approaches. Therefore, we believe that this work has the potential to benefit the broader LLM reasoning community by offering a theoretical foundation. As this paper is a theoretical study, we do not identify negative societal impacts that need to be discussed.

Acknowledgments and Disclosure of Funding

We appreciate the anonymous reviewers for their valuable insights and helpful comments. This work is supported by the National Natural Science Foundation of China (Grant No. 624B2068 and 62576162), the Leading-edge Technology Program of Jiangsu Science Foundation (BK20232003), the Key Program of Jiangsu Science Foundation (BK20243012), and the Fundamental Research Funds for the Central Universities (022114380023).

References

[1] Yasin Abbasi-Yadkori, Ilja Kuzborskij, András György, and Csaba Szepesvári. To believe or not to believe your LLM. *CoRR*, 2024.

[2] Aman Madaan Pranjal Aggarwal, Yiming Yang, and Mausam. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12375–12396, 2023.

[3] Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *CoRR*, abs/2108.07732, 2021.

[4] Evan Becker and Stefano Soatto. Cycles of thought: Measuring LLM confidence through stable explanations. *CoRR*, abs/2406.03441, 2024.

[5] Abhijit Bendale and Terrance E. Boult. Towards open set deep networks. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1563–1572, 2016.

[6] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(1):993–1022, 2003.

[7] Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. Codet: Code generation with generated tests. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

[8] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021.

[9] Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models, 2025.

[10] Stanley F Chen, Douglas Beeferman, and Roni Rosenfeld. Evaluation metrics for language models. 1998.

[11] Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2311.17311*, 2023.

[12] Furui Cheng, Vilém Zouhar, Simran Arora, Mrinmaya Sachan, Hendrik Strobelt, and Mennatallah El-Assady. Relic: Investigating large language model responses using self-consistency. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–18, 2024.

[13] Yang Deng, Wenxuan Zhang, Wai Lam, See-Kiong Ng, and Tat-Seng Chua. Plug-and-play policy planner for large language model powered dialogue agents. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

[14] Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models. In *Findings of the Association for Computational Linguistics*, pages 3563–3578, 2024.

[15] Jinhao Duan, Hao Cheng, Shiqi Wang, Alex Zavalny, Chenan Wang, Renjing Xu, Bhavya Kailkhura, and Kaidi Xu. Shifting attention to relevance: Towards the predictive uncertainty quantification of free-form large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 5050–5063, 2024.

[16] Meng Fang, Xiangpeng Wan, Fei Lu, Fei Xing, and Kai Zou. Mathodyssey: Benchmarking mathematical problem-solving skills in large language models using odyssey math data. *CoRR*, abs/2406.18321, 2024.

[17] Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

[18] Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. *CoRR*, abs/2508.15260, 2025.

[19] 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, 2025.

[20] Lin Gui, Cristina Garbacea, and Victor Veitch. Bonbon alignment for large language models and the sweetness of best-of-n sampling. In *Advances in Neural Information Processing Systems*, 2024.

[21] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

[22] Trevor Hastie, Jerome H. Friedman, and Robert Tibshirani. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, 2001.

[23] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 3828–3850, 2024.

[24] Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with APPS. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

[25] 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. In *Advances in Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

[26] Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiaxin Huang. Efficient test-time scaling via self-calibration. *CoRR*, abs/2503.00031, 2025.

[27] Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiaxin Huang. Efficient test-time scaling via self-calibration. *CoRR*, abs/2503.00031, 2025.

[28] Yuheng Huang, Jiayang Song, Zhijie Wang, Shengming Zhao, Huaming Chen, Felix Juefei-Xu, and Lei Ma. Look before you leap: An exploratory study of uncertainty measurement for large language models. *arXiv preprint arXiv:2307.10236*, 2023.

[29] Yuu Jinnai, Tetsuro Morimura, Kaito Ariu, and Kenshi Abe. Regularized best-of-n sampling to mitigate reward hacking for language model alignment. *CoRR*, abs/2404.01054, 2024.

[30] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know. *CoRR*, abs/2207.05221, 2022.

[31] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, pages 22199–22213, 2022.

[32] László Kozma. Useful inequalities, 2021.

[33] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*, pages 3843–3857, 2022.

[34] Chengpeng Li, Zheng Yuan, Hongyi Yuan, Guanting Dong, Keming Lu, Jiancan Wu, Chuanqi Tan, Xiang Wang, and Chang Zhou. MuggleMath: Assessing the impact of query and response augmentation on math reasoning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 10230–10258, 2024.

[35] Chengshu Li, Jacky Liang, Andy Zeng, Xinyun Chen, Karol Hausman, Dorsa Sadigh, Sergey Levine, Li Fei-Fei, Fei Xia, and Brian Ichter. Chain of code: Reasoning with a language model-augmented code emulator. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[36] Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

[37] Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In *Proceddings of the 12th International Conference on Learning Representations*, 2024.

[38] Zenan Li, Zhi Zhou, Yuan Yao, Yu-Feng Li, Chun Cao, Fan Yang, Xian Zhang, and Xiaoxing Ma. Neuro-symbolic data generation for math reasoning. In *Advances in Neural Information Processing Systems*, pages 23488–23515, 2024.

[39] Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, pages 3622–3628, 2020.

[40] Yuxuan Liu, Tianchi Yang, Shaohan Huang, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, and Qi Zhang. Calibrating llm-based evaluator. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*, pages 2638–2656, 2024.

[41] Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. In *Proceddings of the 13th International Conference on Learning Representations*, 2025.

[42] Viktor Malík and Tomáš Vojnar. Automatically checking semantic equivalence between versions of large-scale c projects. In *Proceedings of the14th IEEE Conference on Software Testing, Verification and Validation*, pages 329–339, 2021.

[43] A Marazzi and C Ruffieux. The truncated mean of an asymmetric distribution. *Computational Statistics & Data Analysis*, 32(1):79–100, 1999.

[44] Bhuvanashree Murugadoss, Christian Poelitz, Ian Drosos, Vu Le, Nick McKenna, Carina Suzana Negreanu, Chris Parnin, and Advait Sarkar. Evaluating the evaluator: Measuring LLMs' adherence to task evaluation instructions. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence*, 2025.

[45] Siqi Ouyang and Lei Li. Autoplan: Automatic planning of interactive decision-making tasks with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3114–3128, 2023.

[46] Sungjin Park, Xiao Liu, Yeyun Gong, and Edward Choi. Ensembling large language models with process reward-guided tree search for better complex reasoning. *CoRR*, abs/2412.15797, 2024.

[47] 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 *Proceedings of the 1st Conference on Language Modeling*, 2024.

[48] Elena Sblendorio, Vincenzo Dentamaro, Alessio Lo Cascio, Francesco Germini, Michela Piredda, and Giancarlo Cicolini. Integrating human expertise & automated methods for a dynamic and multi-parametric evaluation of large language models' feasibility in clinical decision-making. *International Journal of Medical Informatics*, 188:105501, 2024.

[49] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024.

[50] Amir Taubenfeld, Tom Sheffer, Eran Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and Gal Yona. Confidence improves self-consistency in llms. *CoRR*, abs/2502.06233, 2025.

[51] Amir Taubenfeld, Tom Sheffer, Eran Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and Gal Yona. Confidence improves self-consistency in llms. In *Findings of the Association for Computational Linguistics*, pages 20090–20111, 2025.

[52] Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D. Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5433–5442. Association for Computational Linguistics, 2023.

[53] Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. On the planning abilities of large language models - A critical investigation. In *Advances in Neural Information Processing Systems*, pages 75993–76005, 2023.

[54] Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. Dynamic self-consistency: Leveraging reasoning paths for efficient LLM sampling. *CoRR*, abs/2408.17017, 2024.

[55] Junlin Wang, Siddhartha Jain, Dejiao Zhang, Baishakhi Ray, Varun Kumar, and Ben Athiwaratkun. Reasoning in token economies: Budget-aware evaluation of LLM reasoning strategies. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19916–19939, 2024.

[56] Xinglin Wang, Shaoxiong Feng, Yiwei Li, Peiwen Yuan, Yueqi Zhang, Boyuan Pan, Heda Wang, Yao Hu, and Kan Li. Make every penny count: Difficulty-adaptive self-consistency for cost-efficient reasoning. *CoRR*, abs/2408.13457, 2024.

[57] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *Proceedings of the 11th International Conference on Learning Representations*, 2022.

[58] 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 *Advances in Neural Information Processing Systems*, pages 24824–24837, 2022.

[59] Waloddi Weibull. A statistical distribution function of wide applicability. *Journal of applied mechanics*, 1951.

[60] Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

[61] Ruixin Yang, Dheeraj Rajagopal, Shirley Anugrah Hayati, Bin Hu, and Dongyeop Kang. Confidence calibration and rationalization for LLMs via multi-agent deliberation. In *International Conference on Learning Representations Workshop on Reliable and Responsible Foundation Models*, 2024.

[62] Xiao-Wen Yang, Jie-Jing Shao, Lan-Zhe Guo, Bo-Wen Zhang, Zhi Zhou, Lin-Han Jia, Wang-Zhou Dai, and Yu-Feng Li. Neuro-symbolic artificial intelligence: Towards improving the reasoning abilities of large language models. In *Proceedings of the 34th International Joint Conference on Artificial Intelligence*, pages 10770–10778, 2025.

[63] 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. In *Advances in Neural Information Processing Systems*, pages 11809–11822, 2023.

[64] Yuxuan YAO, Han Wu, Zhijiang Guo, Zhou Biyan, Jiahui Gao, Sichun Luo, Hanxu Hou, Xiaojin Fu, and Linqi Song. Learning from correctness without prompting makes LLM efficient reasoner. In *Proceedings of the 1st Conference on Language Modeling*, 2024.

[65] Huaiyuan Ying, Shuo Zhang, Linyang Li, Zhejian Zhou, Yunfan Shao, Zhaoye Fei, Yichuan Ma, Jiawei Hong, Kuikun Liu, Ziyi Wang, Yudong Wang, Zijian Wu, Shuaibin Li, Fengzhe Zhou, Hongwei Liu, Songyang Zhang, Wenwei Zhang, Hang Yan, Xipeng Qiu, Jiayu Wang, Kai Chen, and Dahua Lin. Internlm-math: Open math large language models toward verifiable reasoning, 2024.

[66] 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. In *Proceddings of the 12th International Conference on Learning Representations*, 2024.

[67] Parvez Zamil and Gollam Rabby. Aime problems 1983 to 2024, 2024.

[68] Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. ReST-MCTS*: LLM self-training via process reward guided tree search. In *Proceedings of the 38th Annual Conference on Neural Information Processing Systems*, 2024.

[69] Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics*, pages 10495–10516, 2025.

[70] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

[71] Xinran Zhao, Hongming Zhang, Xiaoman Pan, Wenlin Yao, Dong Yu, Tongshuang Wu, and Jianshu Chen. Fact-and-reflection (far) improves confidence calibration of large language models. In *Findings of the Association for Computational Linguistics*, pages 8702–8718, 2024.

[72] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have claimed that our contributions explicitly at the end of introduction section.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have discussed the limitations at the end of conclusion section as well as in the Appendix E.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We have provided the full set of assumptions in our theoretical results and a complete proof in the Appendix A.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: All the necessary information for reproducibility is provided in the subsection 4.1 and Appendix C. Moreover, we provide all necessary information in our project homepage `https://wnjxyk.github.io/RPC`.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Yes, the code are provided in our supplementary material.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All the necessary information is provided in Section 4.1 and Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We reported the standard deviation of the experiments in the main paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have provided the details of the compute resources in Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors have reviewed the NeurIPS Code of Ethics and the paper conforms to the code of ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We have discussed the potential positive and negative societal impacts of our work in the Section 6.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We have not released any models or datasets that have a high risk for misuse.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have properly credited the creators of the assets in our reference.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: Our paper does not introduce any new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our paper does not involve research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: Our algorithm is confidence estimation method based on the sampled LLM reasoning paths.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Theoretical Results

A.1 Proof of Proposition 1, Proposition 2 and Proposition 3

Proof. In this proof, all the expectations are taken over the combinations of n-sampled reasoning paths $\tilde{t}_1, \ldots, \tilde{t}_n$ and we omit the details of expectation for simplicity. We first prove the general error decomposition in Proposition 1. Then, we give the proofs of Proposition 2 and Proposition 3. We adopt a common and practical assumption that the LLM sampling can be modeled as sampling from a Bernoulli distribution [55], which allows us to further compute the estimation error for specific methods.

Proof of General Error Decomposition For any input x with ground-truth answer y and any possible answer $\hat{y}$, let $\hat{p}(\hat{y} \mid x)$ denote the unbiased estimated confidence of $\hat{y}$ and $p(\hat{y} \mid x)$ denote the ground truth confidence. We have

$$\begin{aligned}
\mathcal{E}_{\hat{p}}(\hat{y}) &= \mathbb{E}\left[\big(\hat{p}(\hat{y} \mid x) - \mathbb{I}[\hat{y} = y]\big)^2\right] \\
&= \mathbb{E}\left[\big(\hat{p}(\hat{y} \mid x) - p(\hat{y} \mid x) + p(\hat{y} \mid x) - \mathbb{I}[\hat{y} = y]\big)^2\right] \\
&= \underbrace{\mathbb{E}\left[\big(\hat{p}(\hat{y} \mid x) - p(\hat{y} \mid x)\big)^2\right]}_{\text{Estimation Error}} + \underbrace{\big(p(\hat{y} \mid x) - \mathbb{I}[\hat{y} = y]\big)^2}_{\text{Model Error}} \\
&\quad + \underbrace{2\mathbb{E}\left[\big(\hat{p}(\hat{y} \mid x) - p(\hat{y} \mid x)\big)\big(p(\hat{y} \mid x) - \mathbb{I}[\hat{y} = y]\big)\right]}_{\text{Cross Term}}
\end{aligned} \tag{11}$$

Then, we prove the Cross Term is zero for the unbiased confidence function as follows:

$$\begin{aligned}
\text{Cross Term} &= 2\big(p(\hat{y} \mid x) - \mathbb{I}[\hat{y} = y]\big) \cdot \mathbb{E}\left[\hat{p}(\hat{y} \mid x) - p(\hat{y} \mid x)\right] \\
&= 2\big(p(\hat{y} \mid x) - \mathbb{I}[\hat{y} = y]\big) \cdot 0 \\
&= 0
\end{aligned} \tag{12}$$

Therefore, we have the general error decomposition:

$$\mathcal{E}_{\hat{p}}(\hat{y}) = \underbrace{\mathbb{E}\left[\big(\hat{p}(\hat{y} \mid x) - p(\hat{y} \mid x)\big)^2\right]}_{\text{Estimation Error}} + \underbrace{\big(p(\hat{y} \mid x) - \mathbb{I}[\hat{y} = y]\big)^2}_{\text{Model Error}} \tag{13}$$

Proof of Error Decomposition for SC For any input x with ground-truth answer y, let $\hat{p}^{(\text{SC})}(\hat{y} \mid x)$ denote the estimated probability of $\hat{y}$ by SC method for any possible answer $\hat{y}$. The confidence function of SC is $\hat{p}^{(\text{SC})}(\hat{y} \mid x) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}[\tilde{y}_i = \hat{y}_i]$, where $\tilde{y}_1 = g(\tilde{t}_1), \ldots, \tilde{y}_n = g(\tilde{t}_n)$ and $\tilde{t}_1, \ldots, \tilde{t}_n$ are $n-$sampled reasoning paths from the LLM distribution. Apply the error decomposition, we have

$$\begin{aligned}
\mathcal{E}_{\hat{p}^{(\text{SC})}}(\hat{y}) &= \mathbb{E}\left[\big(\hat{p}^{(\text{SC})}(\hat{y} \mid x) - p(\hat{y} \mid x)\big)^2\right] + \big(p(\hat{y} \mid x) - \mathbb{I}[\hat{y} = y]\big)^2 \\
&= \frac{1}{n} p(\hat{y} \mid x)(1 - p(\hat{y} \mid x)) + \big(p(\hat{y} \mid x) - \mathbb{I}[\hat{y} = y]\big)^2 .
\end{aligned} \tag{14}$$

Proof of Error Decomposition for PPL Another way is to use the confidence function to build the sampling probability distribution using the unique set of n sampled reasoning paths $\mathcal{R} = \text{Set}\left(\tilde{t}_1, \ldots, \tilde{t}_n\right)$, i.e.,

$$\hat{p}^{(\text{PPL})}(\hat{t} \mid x) = \begin{cases} p(\tilde{t}_i \mid x), & \text{if there is } \tilde{t}_i = \hat{t} \\ 0, & \text{otherwise} \end{cases} = \sum_{\tilde{t} \in \mathcal{R}} \mathbb{I}(\tilde{t} = \hat{t}) p(\tilde{t} \mid x). \tag{15}$$

To simplify the analysis, we assume that each $\tilde{t}_1, \ldots, \tilde{t}_n$ is different from each other, which is a reasonable assumption in practical LLM reasoning and slightly affects the value of n if this assumption is not satisfied. Then, we have

$$\begin{aligned}
\mathbb{E}\left[\hat{p}^{(\text{PPL})}(\hat{t} \mid x) - p(\hat{t} \mid x)\right] &= -(1 - p(\hat{t} \mid x))^n p(\hat{t} \mid x) \\
\mathbb{E}\left[(\hat{p}^{(\text{PPL})}(\hat{t} \mid x) - p(\hat{t} \mid x))^2\right] &= (1 - p(\hat{t} \mid x))^n p(\hat{t} \mid x)^2 .
\end{aligned} \tag{16}$$

Hence,

$$\begin{aligned}\mathcal{E}_{\hat{p}^{(\text{PPL})}}(\hat{t}) &= \mathbb{E}\big[(\hat{p}^{(\text{PPL})}(\hat{t}\,|\,x) - \mathbb{I}[\hat{y}=y])^2\big] \\ &= \mathbb{E}\big[(\hat{p}^{(\text{PPL})}(\hat{t}\,|\,x) - p(\hat{t}\,|\,x) + p(\hat{t}\,|\,x) - \mathbb{I}[g(\hat{t})=y])^2\big] \\ &= -(1-p(\hat{t}\,|\,x))^n p(\hat{t}\,|\,x)^2 + 2(1-p(\hat{t}\,|\,x))^n p(\hat{t}\,|\,x)\mathbb{I}[g(\hat{t})=y] + (p(\hat{t}\,|\,x) - \mathbb{I}[g(\hat{t})=y])^2 \\ &= (1-p(\hat{t}\,|\,x))^n p(\hat{t}\,|\,x)(2\mathbb{I}[g(\hat{t})=y] - p(\hat{t}\,|\,x)) + (p(\hat{t}\,|\,x) - \mathbb{I}[g(\hat{t})=y])^2.\end{aligned} \tag{17}$$

Hence, we complete the proof. □

A.2 Model Error Comparison in Ideal Case

Proposition 4 (Comparison of Model Errors)**.** *Consider a setting with infinite sampling of reasoning paths ($n \to \infty$) where each incorrect reasoning path leads to a unique answer, that is, $g(\tilde{t}_i) \neq g(\tilde{t}_j)$ for any $i \neq j$ where $g(\tilde{t}_i) \neq y$ and $g(\tilde{t}_j) \neq y$. The model error of self-consistency ($\mathcal{E}^{(\text{SC})}_{Model}$) and perplexity ($\mathcal{E}^{(\text{PPL})}_{Model}$) satisfy:*

$$\mathcal{E}^{(\text{SC})}_{Model} \leq \mathcal{E}^{(\text{PPL})}_{Model} \tag{18}$$

The inequality is strict when the consistency function identifies equivalent correct reasoning paths more than once.

Remark 9. Although the assumptions in Proposition 4 are idealized, this special case demonstrates that the consistency function in self-consistency achieves better model error than the perplexity method. In practice, the perplexity method always gives larger model error compared to the self-consistency method, as it does not leverage the consistency function to analyze the structural properties of specific reasoning problems. The proof is presented as follows.

Proof. We first recall the definitions of the model error for self-consistency and perplexity:

$$\begin{cases}\mathcal{E}^{(\text{SC})}_{\text{Model}} & = \sum_{\hat{y}\in\{g(\tilde{t}_i)\,|\,i=1\ldots n\}} \left(\hat{p}^{(\text{SC})}(\hat{y}\,|\,x) - \mathbb{I}[\hat{y}=y]\right)^2, \\ \mathcal{E}^{(\text{PPL})}_{\text{Model}} & = \sum_{\hat{t}\in\mathcal{R}} \left(\hat{p}^{(\text{PPL})}(\hat{t}\,|\,x) - \mathbb{I}[g(\hat{t})=y]\right)^2.\end{cases} \tag{19}$$

where $\mathcal{R} = \text{Set}(\tilde{t}_1, \ldots, \tilde{t}_n)$ is the set of reasoning paths sampled from the LLM. We can compute the difference between the model error of SC and PPL as follows:

$$\begin{aligned}\mathcal{E}^{(\text{SC})}_{\text{Model}} - \mathcal{E}^{(\text{PPL})}_{\text{Model}} &= \sum_{\hat{y}\in\{g(\tilde{t}_i)\,|\,i=1\ldots n\}} \left(\hat{p}^{(\text{SC})}(\hat{y}\,|\,x) - \mathbb{I}[\hat{y}=y]\right)^2 - \sum_{\hat{t}\in\mathcal{R}} \left(\hat{p}^{(\text{PPL})}(\hat{t}\,|\,x) - \mathbb{I}[g(\hat{t})=y]\right)^2 \\ &= \sum_{\hat{y}\in\{g(\hat{t}_i)\,|\,i=1\ldots n\}} \underbrace{\left[\left(\hat{p}^{(\text{SC})}(\hat{y}\,|\,x) - \mathbb{I}[\hat{y}=y]\right)^2 - \sum_{\hat{t}\in\mathcal{R}} \mathbb{I}[g(\hat{t})=\hat{y}] \left(\hat{p}^{(\text{PPL})}(\hat{t}\,|\,x) - \mathbb{I}[\hat{y}=y]\right)^2\right]}_{(A)}\end{aligned} \tag{20}$$

Assuming infinite samplings, the unbiasedness of SC gives us:

$$\hat{p}^{(\text{SC})}(\hat{y}\,|\,x) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}[g(\tilde{t}_i)=\hat{y}] = \sum_{\hat{t}\in\mathcal{R}} \mathbb{I}[g(\hat{t})=\hat{y}] \cdot \hat{p}^{(\text{PPL})}(\hat{t}\,|\,x). \tag{21}$$

For any $\hat{y}$, consider two cases:

(i) If $\hat{y} = y$, then $\hat{y}$ is the correct answer. We have

$$\begin{aligned}
(A) &= \left(\hat{p}^{(\text{SC})}(\hat{y} \mid x) - 1\right)^2 - \sum_{\hat{t} \in \mathcal{R}} \mathbb{I}[g(\hat{t}) = \hat{y}] \left(\hat{p}^{(\text{PPL})}(\hat{t} \mid x) - 1\right)^2 \\
&= \left(\hat{p}^{(\text{SC})}(\hat{y} \mid x)^2 + 1 - 2\hat{p}^{(\text{SC})}(\hat{y} \mid x)\right) - \sum_{\hat{t} \in \mathcal{R}} \mathbb{I}[g(\hat{t}) = \hat{y}] \left(\hat{p}^{(\text{PPL})}(\hat{t} \mid x)^2 + 1 - 2\hat{p}^{(\text{PPL})}(\hat{t} \mid x)\right) \\
&= \hat{p}^{(\text{SC})}(\hat{y} \mid x)^2 + 1 - \sum_{\hat{t} \in \mathcal{R}} \mathbb{I}[g(\hat{t}) = \hat{y}] \cdot \hat{p}^{(\text{PPL})}(\hat{t} \mid x)^2 - \sum_{\hat{t} \in \mathcal{R}} \mathbb{I}[g(\hat{t}) = \hat{y}] \\
&\leq \hat{p}^{(\text{SC})}(\hat{y} \mid x)^2 + 1 - \frac{\hat{p}^{(\text{SC})}(\hat{y} \mid x)^2}{\sum_{\hat{t} \in \mathcal{R}} \mathbb{I}[g(\hat{t}) = \hat{y}]} - \sum_{\hat{t} \in \mathcal{R}} \mathbb{I}[g(\hat{t}) = \hat{y}]
\end{aligned} \tag{22}$$

Let $\hat{p}^{(\text{SC})}(\hat{y} \mid x)^2 = B^2$ and $\sum_{\hat{t} \in \mathcal{R}} \mathbb{I}[g(\hat{t}) = \hat{y}] = C$, then

$$\begin{aligned}
(A) &\leq B^2 + 1 - \frac{B^2}{C} - C \\
&= \frac{1}{C}\left[B^2 C + C - B^2 - C^2\right] \\
&= \frac{1}{C}\left[(C - B^2)(1 - C)\right] \\
&\leq 0.
\end{aligned} \tag{23}$$

Equality holds if $\sum_{\hat{t} \in \mathcal{R}} \mathbb{I}[g(\hat{t}) = \hat{y}] = C = 1$. This indicates that $(A) < 0$ if the consistency function is effective at least once, making $\sum_{\hat{t} \in \mathcal{R}} \mathbb{I}[g(\hat{t}) = \hat{y}] = C > 1$.

(ii) If $\hat{y} \neq y$, then $\hat{y}$ is incorrect. Assuming distinct answers for wrong reasoning paths, we have $\sum_{\hat{t} \in \mathcal{R}} \mathbb{I}[g(\hat{t}) = \hat{y}] = 1$, thus

$$\begin{aligned}
(A) &= \left[\left(\hat{p}^{(\text{SC})}(\hat{y} \mid x) - \mathbb{I}[\hat{y} = y]\right)^2 - \sum_{\hat{t} \in \mathcal{R}} \mathbb{I}[g(\hat{t}) = \hat{y}] \left(\hat{p}^{(\text{PPL})}(\hat{t} \mid x) - \mathbb{I}[\hat{y} = y]\right)^2\right] \\
&= \left[\left(\hat{p}^{(\text{SC})}(\hat{y} \mid x) - \mathbb{I}[\hat{y} = y]\right)^2 - \left(\hat{p}^{(\text{SC})}(\hat{y} \mid x) - \mathbb{I}[\hat{y} = y]\right)^2\right] = 0,
\end{aligned} \tag{24}$$

since only one $\hat{t} \in \mathcal{R}$ satisfying that $g(\hat{t})$ equals the incorrect answer $\hat{y}$.

Therefore, $\mathcal{E}^{(\text{PPL})}_{\text{Model}} - \mathcal{E}^{(\text{SC})}_{\text{Model}} \leq 0$, indicating that the model error of self-consistency is always less than or equal to the model error of perplexity under our assumptions. Moreover, if the consistency function is effective at least once, the model error of self-consistency is strictly less than the model error of perplexity. □

A.3 Proof of Theorem 4

Proof. In this proof, all the expectations are taken over the combinations of n-sampled reasoning paths $\tilde{t}_1, \ldots, \tilde{t}_n$ for estimating the confidence $\hat{p}^{(\text{PC})}$ and we omit the details of expectation for simplicity. To simplify the analysis, we assume that each $\tilde{t}_1, \ldots, \tilde{t}_n$ is different from each other, which is a reasonable assumption in practical LLM reasoning and slightly affects the value of n if this assumption is not satisfied. For any given answer $\hat{y}$, the estimated probability of PC is $\hat{p}(\hat{y} \mid x) = \sum_{i=1}^{n} \mathbb{I}[g(\tilde{t}_i) = \hat{y}] p(\tilde{t}_i \mid x)$, allowing the reasoning error of PC can be computed as follows.

$$\begin{aligned}
\mathcal{E}_{\hat{p}^{(\text{PC})}}(\hat{y}) &= \mathbb{E}\left[(\hat{p}^{(\text{PC})}(\hat{y} \mid x) - \mathbb{I}[\hat{y} = y])^2\right] \\
&= \mathbb{E}\left[(\hat{p}^{(\text{PC})}(\hat{y} \mid x) - p(\hat{y} \mid x) + p(\hat{y} \mid x) - \mathbb{I}[\hat{y} = y])^2\right].
\end{aligned} \tag{25}$$

We define $k := |\{\tilde{t} \mid g(\tilde{t}) = \hat{y}\}|$, which means that how many reasoning paths whose answers are $\hat{y}$ can be covered given limited sampling budget. Note that we further have $\mathbb{E}[\mathbb{I}[g(\tilde{t}) = \hat{y}] p(\tilde{t} \mid x)] =$

$\frac{1}{k}p(\hat{y}\,|\,x)$, thus

$$\begin{aligned}\mathbb{E}[\hat{p}^{(\text{PC})}(\hat{y}\,|\,x)] &= \mathbb{E}\big[\sum_{i=1}^{n}\mathbb{I}[g(\tilde{t}_i)=\hat{y}]p(\tilde{t}_i\,|\,x)\big]\\ &= \Big(1-\big(1-\frac{1}{k}p(\hat{y}\,|\,x)\big)^n\Big)p(\hat{y}\,|\,x)\\ &= (1-\alpha^n)p(\hat{y}\,|\,x),\end{aligned} \tag{26}$$

where $\alpha := \frac{1}{k}p(\hat{y}\,|\,x)$. Again, we have

$$\begin{aligned}\mathbb{E}[\hat{p}^{(\text{PC})}(\hat{y}\,|\,x) - p(\hat{y}\,|\,x)] &= -\big(1-\alpha\big)^n p(\hat{y}\,|\,x))\\ \mathbb{E}[(\hat{p}^{(\text{PC})}(\hat{y}\,|\,x) - p(\hat{y}\,|\,x))^2] &= \Big(1-\big(1-\alpha\big)^n\Big)\big(1-\alpha\big)^n p(\hat{y}\,|\,x))^2\end{aligned} \tag{27}$$

Hence,

$$\begin{aligned}\mathcal{E}_{\hat{p}^{(\text{PC})}}(\hat{y}) &= \mathbb{E}\big[(\hat{p}^{(\text{PC})}(\hat{y}\,|\,x) - \mathbb{I}[\hat{y}=y])^2\big]\\ &= \mathbb{E}\big[(\hat{p}^{(\text{PC})}(\hat{y}\,|\,x) - p(\hat{y}\,|\,x) + p(\hat{y}\,|\,x) - \mathbb{I}[\hat{y}=y])^2\big]\\ &= (1-\alpha)^n p(\hat{y}\,|\,x)\big(2\mathbb{I}[\hat{y}=y] - (1+(1-\alpha)^n)p(\hat{y}\,|\,x)\big) + \big(p(\hat{y}\,|\,x) - \mathbb{I}[\hat{y}_i=y]\big)^2,\end{aligned} \tag{28}$$

which completes the proof. □

A.4 Proof of Theroem 7

Proof. Let us set the pruning threshold by $\tau := p(y\,|\,x)$. Then, we have

$$\begin{aligned}\mathcal{E}_{\hat{p}^{(\text{RPC})}}(\hat{y}) = &\underbrace{\alpha p(\hat{y}\,|\,x)\big(2\mathbb{I}[\hat{y}=y] - (1+\alpha)p(\hat{y}\,|\,x)\big)\mathbb{I}[(p(\hat{y})\,|\,x)<\tau]}_{\text{Estimation Error}}\\ &+\underbrace{\big(p(\hat{y}\,|\,x) - \mathbb{I}[\hat{y}_i=y]\big)^2\,\mathbb{I}[(p(\hat{y})\,|\,x)<\tau]}_{\text{Model Error}}\end{aligned} \tag{29}$$

However, we only have an estimation of $p(\hat{y}\,|\,x)$, i.e., $\hat{p}^{(\text{RPC})}(\hat{y}\,|\,x) = k\mathbb{E}\left[\mathbb{I}[g(\tilde{t})=\hat{y}]p(\tilde{t}\,|\,x)\right] \approx \frac{k}{\hat{k}}\sum_{i=1}^{k}p(\tilde{t}_i\,|\,x)$, where $\tilde{t}_1,\ldots,\tilde{t}_{\hat{k}}$ are $\hat{k}$ sampled reasoning paths whose answer is $\hat{y}$. Therefore, the reasoning error of our approximate version can be computed by

$$\begin{aligned}\mathcal{E}_{\hat{p}^{(\text{RPC})}}(\hat{y}) = &\underbrace{\alpha(p)p(\hat{y}\,|\,x)\big(2\mathbb{I}[\hat{y}=y] - (1+\alpha(p))p(\hat{y}\,|\,x)\big)\mathbb{I}[\frac{1}{k}\sum_{i=1}^{\hat{k}}p(\tilde{t}_i\,|\,x)<\frac{1}{m}\tau]}_{\text{Estimation Error}}\\ &+\underbrace{\big(p(\hat{y}\,|\,x) - \mathbb{I}[\hat{y}_i=y]\big)^2\,\mathbb{I}[\frac{1}{\hat{k}}\sum_{i=1}^{\hat{k}}p(\tilde{t}_i\,|\,x)<\frac{1}{m}\tau]}_{\text{Model Error}}\end{aligned} \tag{30}$$

Hence, we only need to consider the probability $\frac{1}{\hat{k}}\sum_{i=1}^{\hat{k}}p(\tilde{t}_i\,|\,x) > \frac{1}{m}\tau$. Using Hoeffding's inequality, we can obtain that

$$\mathbb{P}\big(\frac{1}{\hat{k}}\sum_{i=1}^{\hat{k}}p(\tilde{t}_i\,|\,x) - \frac{1}{k}p(\hat{y}\,|\,x) \geq \tau\big) \leq \exp\Big(-\frac{2\hat{k}\gamma^2}{p(\hat{y}\,|\,x)^2}\Big) \tag{31}$$

We set $\gamma = \tau - \frac{1}{k}p(\hat{y}\,|\,x) = \tau + \alpha - 1$, then

$$\mathbb{P}\big(\frac{1}{\hat{k}}\sum_{i=1}^{\hat{k}}p(\tilde{t}_i\,|\,x) \geq \tau\big) \leq \exp\Big(-2\hat{k}k^2(1-\frac{\tau}{1-\alpha})^2\Big). \tag{32}$$

Hence, we complete the proof. □

B Pseudo Code of RPC Method

In this section, we provide the pseudo code of RPC. The output of Algorithm 1 is a set of reasoning paths with the highest confidence. The extraction function $g(\cdot)$ can be used to parse the reasoning paths to answers.

Algorithm 1 Reasoning-pruning Perplexity Consistency

Input:
Sampled Reasoning paths $\tilde{t}_1, \ldots, \tilde{t}_n$
LLM Internal Probabilities $p_1, \ldots, p_n$
Consisitency function $\mathbb{I}_C(\cdot, \cdot)$
Output: Most-confident reasoning paths with probabilities
{**Phase 1: Reasoning Pruning**}
$(k_1, \lambda_1, k_2, \lambda_2, w_1, w_2) \leftarrow$ Model probability distribution parameters from $p_1, \ldots, p_n$ {Using Theorem 3.2}
$p_{\text{mean}} \leftarrow \frac{1}{n} \sum_{i=1}^{n} p_i$
$\text{I}_{\text{retain}} \leftarrow \{i \mid P_{\text{High}}(p_i) > 0.5 \text{ or } p_i \geq p_{\text{mean}}\}$ {P_{High} is defined in Theorem 3.2}
{**Phase 2: Perplexity Consistency**}
$\text{U} \leftarrow \text{Set}(\tilde{t}_1, \ldots, \tilde{t}_n)$
$\text{I}_{\text{unique}} \leftarrow \{i \mid \tilde{t}_i \in \text{U} \text{ and } i \in \text{I}_{\text{retain}}\}$
for each reasoning path $\tilde{t} \in \text{U}$ **do**
    $\text{C}(\tilde{t}) \leftarrow \sum_{i \in \mathcal{I}_{\text{retain}}} \mathbb{I}_C[\tilde{t}, \tilde{t}_i] p_i$
end for
$\text{C}_{\text{max}} \leftarrow \max_{\tilde{t} \in \text{U}} \text{C}(\tilde{t})$
return $\{(\tilde{t}, \text{C}(\tilde{t})) \mid \tilde{t} \in \text{U}, \text{C}(\tilde{t}) = \text{C}_{\text{max}}\}$

C Detailed Experimental Settings

C.1 Datasets

For mathematical reasoning tasks, we evaluate our proposed methods and comparison methods on four mathematical datasets that include MATH, MathOdyssey, OlympiadBench, and AIME datasets.

- MATH dataset [25] is a dataset comprised of challenging competition math problems and we use its 5,000 testing data for evaluation.
- MathOdyssey dataset [16] contains 387 problems, covering advanced high-school level, university-level, and Olympiad-level mathematics.
- OlympiadBench dataset [23] contains 8,476 Olympiad-level mathematics and physics problems. We select the English problems without images, resulting in a testing dataset of 1,284 problems.
- AIME dataset [67] contains 993 test problems collected from the American Invitational Mathematics Examination, spanning from 1983 to 2024.

For code generation tasks, we conduct experiments on three common benchmark datasets. HumanEval [8] contains 164 hand-written Python programming problems. MBPP [3](sanitized version) consists of 427 entry-level programming problems. We also include the introductory-level problems of APPS [24], which contains 1000 problems.

C.2 Detailes of Mathematical Reasoning Task

For all experiments in the main paper, we use a sampling temperature of 1.0 and set the top-p parameter to 0.95. The prompt template follows the recommendation of each LLM.

Prompt for Math Reasoning Tasks. The InternLM2-MATH-Plus 1.8B and 7B models are chat models that facilitate conversations between two roles: "user" and "assistant". The prompt for the "user" role is provided in Prompt 1. In contrast, the DeepSeek-Math 7B model operates in a non-chat mode, and its corresponding prompt is shown in Prompt 2.

Prompt 1: Prompt for InternLM-2-Math-Plus

Problem:
{instruction}
Let's think step by step
Solution:

Prompt 2: Prompt for DeepSeek-Math

{instruction}
Please reason step by step, and put your final answer within \boxed{}.

Prompt for Mathematical Verbalized Method. We observed that the tuned math models are challenging to prompt for generating confidence. Therefore, we adopted the methods from [52] to calculate the probability based on the likelihood of the first generated "True" token and the first generated "False" token. The corresponding prompt is provided in Prompt 3.

Prompt 3: Prompt for Mathematical Verbalized Method

Question: {question}
Proposed Answer: {answer}
Is the proposed answer:
\t(A) True or
\t(B) False?
The proposed answer is:

C.3 Detailes of Code Generation Task

Code Generation. On the code generation task, we let LLM generate a code snippet to solve a given programming problem, and then evaluate its functional correctness based on the ground-truth test cases provided by the dataset. In detail, we set the top *p* to 0.95, and the max generation length to 1024. For code snippet post-processing, we first extract the code text from the code block surrounded by triple-backticks(‘‘‘), and then we follow [8] to truncate the generated code snippet before the following stop sequences: "\nclass", "\ndef", "\n#", "\nif", "\nprint". At the same time, we also obtain the log-probability of each token from the LLM response. For "verbalization" setting, the verbalized confidence is also extracted from the text generated by LLM along with the code snippet.

Self-consistency on Code. We follow [7] to sample 100 test cases for each programming problem from the same model. Then, we achieved self-consistency in code at the semantic equivalence level, which is based on the execution behavior of any two codes on this set of test cases. More formally, we implemented the consistency function $\mathbb{I}_C(\cdot, \cdot)$ as an indicator function that indicates whether two codes are semantically equivalent, i.e., $\mathbb{I}_C(x, y) = 1$ if and only if code x and y execute the same result on this set of test cases.

Prompt for Generating Code. The prompt for generating code consists of a header, a functional signature, and a docstring and LLM needs to implement the body of this function. An Illustration of `has_close_elements` problem is shown in Prompt 4.

Prompt for Generating Test Cases. For generating test cases, we implement the function body with a "pass" statement on the basis of the prompt to generate the code, and added a comment to require the LLM to generate test cases for the programming problem. An Illustration of `has_close_elements` problem is shown in Prompt 5.

Prompt 4: Prompt for Generating Code

```
def has_close_elements(numbers: List[float],
    threshold: float) -> bool:
"""Check if in given list of numbers, are any two numbers
closer to each other than given threshold.
"""
```

Prompt 5: Prompt for Generating Test Cases.

```
def has_close_elements(numbers: List[float],
    threshold: float) -> bool:
""" Check if in given list of numbers, are any two numbers
closer to each other than given threshold.
"""
    pass
# check the correctness of has_close_elements
assert
```

Prompt for Code Verbalized Method. For generating code with verbalized confidence, we add instructions for generating verbalized confidence, as well as format requirements to facilitate the extraction of code and confidence score. We also give a simple example to help LLM understand the format requirements at the end of the prompt. An Illustration of `has_close_elements` problem is shown in Prompt 6.

Prompt 6: Prompt for Code Verbalized Method

Come up with a solution that solves the following programming question and provide your confidence score in this solution like 0%, 10%, ... 100%.

```
def has_close_elements(numbers: List[float],
    threshold: float) -> bool:
"""Check if in given list of numbers, are any two numbers
closer to each other than given threshold.
"""
```

Format requirement: output in the form of the following example. Do not provide any additional explanations. Here is an output example:
Solution:

````
```python
your code ...
```
````

Confidence:

D Detailed Experimental Results

D.1 Results under High Sampling Temperature.

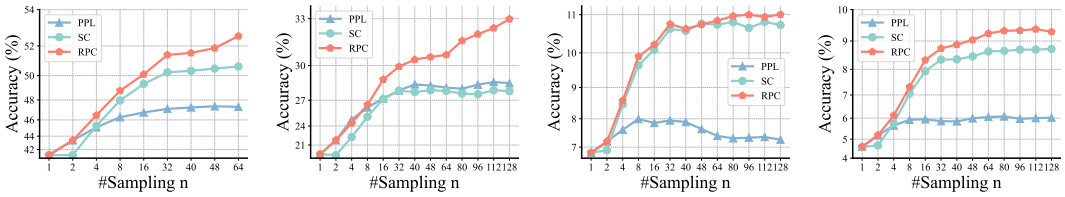
Using a high sampling temperature enables LLMs to produce more diverse outputs, potentially enhancing reasoning performance. However, it also leads to an increase in estimation error.

To investigate the effectiveness of our approaches in addressing the estimation error issue, we conducted experiments with higher sampling temperatures (T = 1.1 and T = 1.3) using the InternLM-2-MATH-Plus 7B model. The results in Table 4 indicate that our RPC approach consistently surpasses baseline methods. Notably, a significant performance gap persists between RPC and SC, indicating that RPC effectively tackles the estimation error issue even under sampling high-temperature.

We also report the performance of RPC with different sampling numbers under the high temperature in Figure 5 and Figure 6. The results demonstrate that the RPC method effectively improves the reasoning performance at higher temperatures, as it leverages the increased diversity in sampling to enhance self-consistency. In contrast, the SC method's performance deteriorates due to increased estimation errors at higher temperatures.

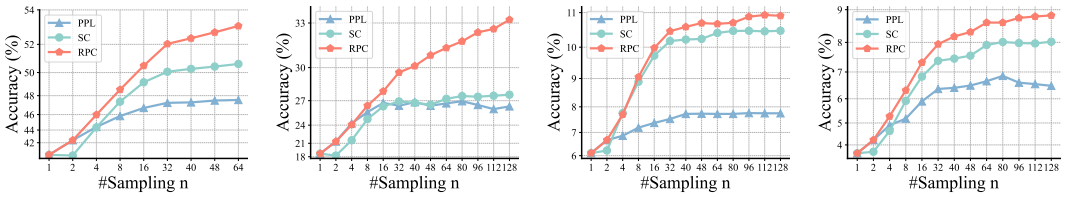
Table 4: Performance Comparison of different models and different parameter scales. The accuracy is reported as the mean and stdev. The best performance is highlighted in **bold**. The results show that our RPC approach consistently outperforms existing methods.

Method	**Temperature = 1.1**			
	MATH	MathOdyssey	OlympiadBench	AIME
PPL	47.35 ± 0.16	28.59 ± 1.30	7.27 ± 0.23	6.02 ± 0.34
VERB	25.51 ± 0.23	9.41 ± 0.44	3.66 ± 0.16	3.07 ± 0.15
SC	50.66 ± 0.22	27.89 ± 0.43	10.74 ± 0.15	8.73 ± 0.24
RPC	**52.58 ± 0.14**	**32.98 ± 0.69**	**11.00 ± 0.24**	**9.30 ± 0.29**
Method	**Temperature = 1.3**			
	MATH	MathOdyssey	OlympiadBench	AIME
PPL	47.58 ± 0.31	26.38 ± 1.41	7.76 ± 0.46	6.50 ± 0.41
VERB	24.62 ± 0.33	8.60 ± 0.26	3.11 ± 0.17	2.29 ± 0.12
SC	50.65 ± 0.14	27.61 ± 0.67	10.49 ± 0.18	8.02 ± 0.20
RPC	**53.12 ± 0.14**	**33.19 ± 0.56**	**10.91 ± 0.18**	**8.83 ± 0.23**



(a) MATH (b) MathOdyssey (c) OlympiadBench (d) AIME

Figure 5: The accuracy of InternLM-2-MATH-Plus 7B model on four mathematical reasoning datasets with different sample sizes n. The sampling temperature is set to 1.1.

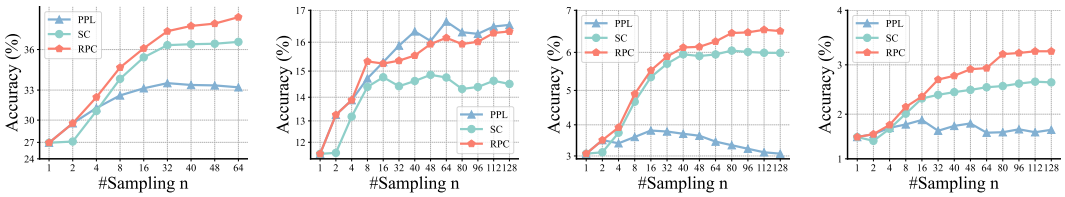


(a) MATH (b) MathOdyssey (c) OlympiadBench (d) AIME

Figure 6: The accuracy of InternLM-2-MATH-Plus 7B model on four mathematical reasoning datasets with different sample sizes n. The sampling temperature is set to 1.3.

D.2 Performance with Different Models Scales

In the Figure 2, we plot the accuracy of the InternLM-2-MATH-Plus 7B model on four mathematical reasoning datasets with different sample sizes n. Here, we further investigate the performance of relateively small model, InternLM-2-MATH-Plus 1.8B, is presented in Figure 7. Similar conclusions can be drawn from these results. For the MathOdyssey dataset, the PPL method shows superior performance compared to other methods, which can be attributed to the relatively low model error of PPL on this dataset, allowing the perplexity-based approach to function effectively. Furthermore, the RPC method consistently outperforms the SC method, which demonstrates its ability to enhance the convergence properties of the SC method.



(a) MATH (b) MathOdyssey (c) OlympiadBench (d) AIME

Figure 7: The accuracy of InternLM-2-MATH-Plus 1.8B model on four mathematical reasoning datasets with different sample sizes n.

D.3 Performance with R1 LLMs

We further investigate the performance of each method using R1 LLMs with thinking capability [9]. Here, we conduct experiments on the DeepSeek-R1-Distill-Qwen-7B model [21] using 16 samples, where the internal probability of LLM is computed only using the response after the "</think>" token.

The results are presented in Table 5. We observe that our RPC method consistently outperforms PPL and SC across all datasets when using R1 LLMs. Moreover, the performance improvements persist even when the base LLM becomes stronger, demonstrating the potential of RPC to improve even stronger LLMs and its generalizability across diverse LLMs.

Table 5: Performance comparison on DeepSeek-R1-Distill-Qwen-7B model. The best performance is highlighted in **bold**. The results show that our RPC is compatible with thinking LLMs and consistently outperforms PPL and SC across all datasets.

Method	MathOdyssey	AIME	MATH	OlympiadBench
PPL	60.04	72.92	81.81	21.65
SC	57.22	70.40	82.03	21.93
RPC	**61.11**	**76.47**	**82.78**	**22.81**

D.4 Performance Comparison with Advanced Methods

In our main paper, we compare RPC with SC and PPL to demonstrate its effectiveness in bridging internal probability and self-consistency, which confirms our theoretical analysis. Here, we further combine RPC with two advanced methods: ESC [36] (an advanced version of SC) and Best-of-N (BoN) using the Qwen2.5-Math-PRM-7B reward model (RM) [69] (an advanced version of PPL that replaces internal probability with rewards from an external reward model) to verify the compatibility of RPC with state-of-the-art methods. Similar to Appendix D.3, we conduct experiments on the DeepSeek-R1-Distill-Qwen-7B model using 16 samples.

As shown in Table 6, RPC using ESC consistently outperforms both SC and ESC across all datasets, demonstrating its effectiveness when adapted to advanced methods. Similarly, the results in Table 7 show that RPC using RM consistently performs better than or comparable to SC and BoN using RM across all datasets. Overall, these experimental results demonstrate that RPC is compatible with state-of-the-art methods and consistently outperforms them across all datasets.

Table 6: Performance comparison with ESC method. The best performance is highlighted in **bold**.

Method	MathOdyssey	AIME	MATH	OlympiadBench
SC	57.22	70.40	82.03	21.93
ESC	57.17	70.18	81.97	22.03
RPC using ESC	**61.03**	**76.26**	**82.74**	**22.81**

Table 7: Performance comparison with BoN method using reward model. The best performance is highlighted in **bold**.

Method	MathOdyssey	AIME	MATH	OlympiadBench
SC	57.22	70.40	82.03	21.93
BoN using RM	**58.35**	69.35	81.86	21.57
RPC using RM	**58.35**	**71.04**	**82.16**	**22.22**

D.5 Performance on Additional Reasoning Tasks

In our main paper, we demonstrate the effectiveness of RPC on both mathematical reasoning and code generation tasks. Here, we further conduct experiments on two additional reasoning tasks: GPQA [47] and LogiQA [39]. GPQA is a multiple-choice benchmark covering biology, physics, and chemistry, while LogiQA is a benchmark testing logical reasoning abilities. Similar to Appendix D.3, we conduct experiments on the DeepSeek-R1-Distill-Qwen-7B model using 16 samples.

As shown in Table 8, we observe that our RPC method consistently outperforms PPL and SC on both datasets. These experimental results demonstrate that the RPC method is generalizable to additional reasoning tasks, showing its strong potential for application to diverse real-world reasoning scenarios.

Table 8: Performance comparison on GPQA and LogiQA benchmarks. The best performance is highlighted in **bold**.

Method	GPQA	LogiQA
PPL	41.46	54.36
SC	43.00	56.71
RPC	**44.09**	**58.42**

D.6 Explanation on Hyper-parameters

We would like to emphasize that the RPC method introduces a Reasoning Pruning module that can automatically determine the pruning of reasoning paths without manually setting thresholds. Therefore, it avoids sensitivity issues associated with manual threshold setting and makes our RPC a hyperparameter-free method, which is robust across diverse tasks, including mathematical reasoning, code generation, logical reasoning, etc.

Although our RPC method does not explicitly require hyperparameter tuning, the optimization process of the Reasoning Pruning module still involves some parameter configurations, such as initialization and the range of distribution parameters. Therefore, to evaluate the robustness of RPC, we conducted experiments analyzing the impact of different initialization methods and parameter bounds. Specifically, we tested the method with 10 random seeds under various configurations, including different initialization approaches (Fixed Init vs. Zero Init) and different bounds for the distribution parameters w_1 and w_2. The Fixed Init method and parameter bounds of $[0.2, 0.8]$ represent our default optimization settings for RPC. Results presented in Table 9 demonstrate that RPC maintains robust performance across different configurations. Note that these parameter configurations in our method do not require tuning in practical applications.

Table 9: Performance of RPC with different initialization methods and parameter bounds when automatically modeling internal LLM probability distributions.

	Fixed Init	**Zero Init**
$w \in [0.2, 0.8]$	31.620 ± 0.754	31.183 ± 0.821
$w \in [0.15, 0.85]$	31.748 ± 0.622	32.108 ± 0.685
$w \in [0.1, 0.9]$	31.722 ± 0.577	32.416 ± 0.532

D.7 Explanation on Computational Overhead

In this section, we analyze the computational overhead of RPC compared to SC from both theoretical and practical perspectives.

From a theoretical standpoint, SC requires $O(n^2)$ time complexity for answer equivalence computation, where n is the number of samplings. Our RPC method consists of two components: Reasoning Pruning with $O(k(m^2+n))$ time complexity and Perplexity Consistency with $O(n_p^2+n_p \log n_p+n_p)$ time complexity, where $m = 5$ represents the number of mixture distribution parameters, k denotes the optimization iteration count, and n_p is the number of reasoning paths after pruning ($n_p \leq n$). The overall complexity simplifies to $O(25k + nk + n_p^2) \leq O(25k + nk + n^2)$, introducing only minimal computational overhead compared to SC.

From a practical perspective, experiments on MathOdyssey with 128 samplings show that SC takes 0.006s per question while RPC takes 0.036s. Both processing times are negligible compared to the LLM inference time required for sampling multiple reasoning paths. Moreover, RPC's slight overhead is well-justified by its ability to reduce the required number of samplings while maintaining performance, as demonstrated in Table 1, which addresses the primary computational bottleneck in reasoning tasks.

Overall, RPC provides an excellent computational trade-off, where the computational overhead of RP is exchanged for significant time savings achieved by reducing the number of LLM inferences.

E Limitations and Future Work

In this paper, we introduce a theoretical framework for analysis the sampling-based test-time scaling methods of LLM reasoning and a new method based on our analyses. While our theoretical framework and approach offer several advantages, there remain some limitations as listed as follows:

1. Our theoretical framework is general to analysis sampling-based test-time scaling methods, however, in this paper, we only focus on the analysis on two typical methods as well as RPC.
2. Our proposed method is a post-hoc approach that relies solely on the sampled reasoning paths and selects answers from these samples. This design makes the method straightforward to use, as it does not require any modifications to the LLM architecture or its training process. However, the performance improvements are moderate compared to approaches that involve model training. Although integrating our method with trained LLMs may provide additional performance gains, it remains an open question how to utilize our insights to improve the training process of LLMs.
3. Current design of our method is relatively simple, which leaves considerable room for improvement. For example, the perplexity consistency component could be extended to include alternative probability measures, such as rewards from a reward model. Furthermore, the reasoning pruning module could be improved by incorporating more advanced techniques to increase its effectiveness.

We also provide the following directions for future work:

1. **Analyzing Test-Time Scaling Methods**: Our paper demonstrates that the theoretical framework is versatile enough to analyze two typical sampling-based test-time scaling methods. It can also be readily extended to evaluate other advanced methods derived from these two types, with strong potential to adapt to various test-time scaling strategies.

2. **Exploring Advanced Sampling Strategies**: Our theoretical results indicate that better convergence requires an effective sampling strategy to sample sufficiently diverse reasoning paths, which is not deeply investigated in this paper. Improved sampling strategies could be designed by drawing inspiration from our theoretical framework.
3. **Applying to Diverse Reasoning Tasks**: In our paper, we show that the Rpc method is effective on math, code, and logical reasoning tasks without exploiting task-specific properties. Task-specific methods could be developed based on our theoretical framework to achieve better performance (e.g., by considering the properties of $g(\cdot)$ in different reasoning tasks).