CGES: Confidence-Guided Early Stopping for Efficient and Accurate Self-Consistency

Ehsan Aghazadeh **Ahmad Ghasemi** **Hedyeh Beyhaghi** **Hossein Pishro-Nik**
University of Massachusetts Amherst
{eaghazadeh,aghasemi,hbeyhaghi,pishro}@umass.edu

Abstract

Large language models (LLMs) are often queried multiple times at test time, with predictions aggregated by majority vote. While effective, this *self-consistency* [Wang et al., 2023] strategy requires a fixed number of calls and fails when the correct answer is infrequent. We introduce ***Confidence-Guided Early Stopping (CGES)***, a Bayesian framework that forms posteriors over candidate answers from scalar confidence signals—derived from token probabilities or reward models—and adaptively halts sampling once posterior mass exceeds a threshold. We provide theoretical guarantees in both the ideal case of perfectly calibrated confidences and the realistic regime with noisy confidences. Averaged over five reasoning benchmarks, CGES reduces the average number of calls by **69.4%** (e.g., from 16.0 to 4.9) while maintaining accuracy within **0.06 percentage points** of self-consistency.

1 Introduction

Large language models (LLMs) have achieved remarkable progress across reasoning, problem solving, and open-domain tasks. A common practice to improve reliability is *test-time scaling* [Snell et al., 2025], a family of methods that allocate additional inference-time computation to improve performance. One subset of these methods samples multiple responses and aggregates them into a final prediction. Among the most widely used methods, self-consistency (SC) [Wang et al., 2023] aggregates outputs by majority vote, leveraging the intuition that the most frequent answer across diverse generations is likely to be correct. While simple and effective in many settings, majority-based aggregation suffers from two major shortcomings. First, it assumes that response frequency is a faithful proxy for correctness, which fails in cases where the correct answer appears infrequently. Second, it requires a fixed number of model calls regardless of confidence, leading to substantial inefficiency.

Confidence signals offer an alternative perspective. Instead of depending solely on frequency, one can incorporate confidence scores that capture the model's belief in each response. These scores may be derived from different sources. *Token probabilities* are taken directly from the model's output distribution and reflect how certain the model is about generating each token. *Calibration schemes* adjust these raw probabilities so that they better match the actual likelihood of correctness, turning overconfident or underconfident estimates into more reliable signals. *External reward models* are trained separately, often with human feedback or domain-specific supervision, and provide an independent measure of response quality beyond the model's own probabilities. *Such signals can distinguish between frequent but uncertain answers and rare yet confident ones*.

We propose a *confidence-based Bayesian framework* for test-time scaling. Our framework builds on the idea that incorporating confidence enables robust aggregation and adaptive stopping, where sampling stops once sufficient certainty is reached, reducing cost without sacrificing accuracy. Our approach computes posterior probabilities over candidate answers by treating each response and

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Efficient Reasoning.

its associated confidence as probabilistic evidence. This yields two key advantages over majority voting: (1) when the correct answer is frequent, our method reaches the same conclusion but often stops earlier by exploiting high-confidence signals, improving efficiency; (2) when the correct answer is a minority, our method can still recover it by amplifying the influence of confident predictions, where majority voting fails. Figure 1 illustrates both phenomena on real items: our framework (a) halts after only a few calls when confidence concentrates, and (b) outperforms majority vote when the correct answer is a minority but highly confident. We further formalize this intuition by proving theoretical guarantees under ideal conditions where confidence scores are faithful to the true likelihood of correctness. We then extend the analysis to the more practical case of noisy confidence estimates, where scores may be imperfect reflections of true correctness.

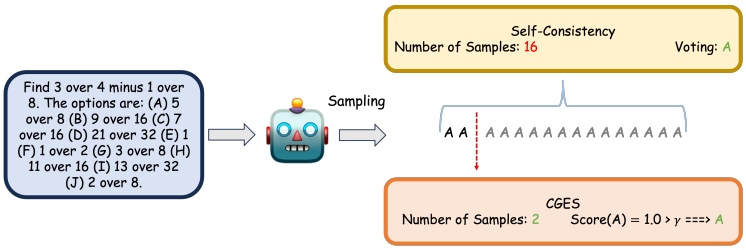
To operationalize this framework, we introduce *Confidence-Guided Early Stopping (CGES)*, which integrates Bayesian scoring with an adaptive stopping rule. CGES allows for flexible accuracy–efficiency trade-offs by halting once posterior concentration exceeds a threshold or a budget is reached. We evaluate CGES across multiple reasoning benchmarks, including AIME24, MATH500 [Hendrycks et al., 2021b], GSM8K [Cobbe et al., 2021], GPQA [Rein et al., 2024], and MMLU_Pro [Wang et al., 2024], and compare against self-consistency [Wang et al., 2023] and early-stopping self-consistency [Li et al., 2024]. Our experiments show that CGES consistently reduces the number of LLM calls by large margins while maintaining or even improving accuracy. These results highlight the benefits of incorporating calibrated confidence into test-time scaling, moving beyond frequency-based heuristics toward principled Bayesian aggregation. In summary, our contributions are as follows:

- We propose a Bayesian framework that incorporates confidence estimates into self-consistency, enabling more accurate and theoretically grounded aggregation beyond majority voting.
- We design *Confidence-Guided Early Stopping (CGES)*, which adaptively halts sampling to trade off accuracy and efficiency.
- We establish theoretical guarantees of correctness under ideal conditions where confidence scores are perfectly calibrated to the true probabilities of correctness, and extend the analysis to the realistic setting with noisy confidence estimates.
- We empirically validate CGES across five reasoning benchmarks, showing substantial efficiency improvements while maintaining or improving accuracy.

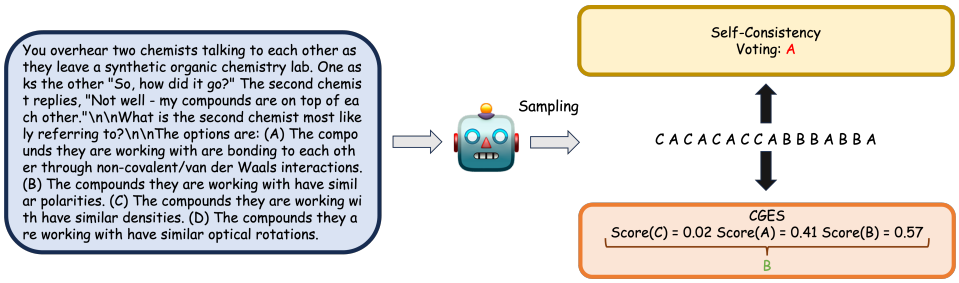
2 Related Work

A widely used approach for improving test-time scaling is *self-consistency*. Introduced by Wang et al. [2023], it aggregates multiple reasoning paths by majority voting, improving reliability in chain-of-thought reasoning. However, self-consistency requires a fixed number of model calls and often fails when the correct answer is infrequent. To reduce this cost, Li et al. [2024] proposed early-stopping self-consistency (ESC), which stops sampling when predictions begin to agree. More recent extensions, such as Self-Calibration [Huang et al., 2025], incorporate dynamic stopping rules or distill self-consistency signals into single-pass confidence estimates. *In contrast to Self-Calibration, which learns confidence from majority voting signals, our work introduces a* ***Bayesian framework*** *with* ***theoretical guarantees*** *for* ***confidence-guided early stopping****.*

Beyond self-consistency, several methods adaptively allocate test-time compute. Snell et al. [2025] study compute-optimal scaling strategies, while Muennighoff et al. [2025] introduce s^1, which optimizes inference length under budget constraints. Other directions combine search and verification: self-enhanced tree search frameworks [Bi et al., 2025, Lample et al., 2022, Koh et al., 2024] expand multiple reasoning paths with sparse activation, while step-wise verifiers dynamically prune the search tree [Li et al., 2023, Lightman et al., 2024]. Two-stage elimination-based approaches [Chen et al., 2025] refine candidate answers iteratively, and query-variant ensembling [Huang et al., 2024] improves robustness. These methods share with us the goal of balancing accuracy and efficiency, but differ in their reliance on structured search or verifier signals. Our approach instead treats confidence as probabilistic evidence in a Bayesian model, yielding lightweight updates and formal consistency guarantees. Moreover, while prior methods typically optimize efficiency when extracting a single reasoning path or answer, our framework belongs to the family of test-time scaling approaches that



(a) Efficiency: CGES stops early once its posterior exceeds γ, while SC uses a fixed budget (B=16).



(b) Accuracy: SC's majority vote is wrong, but CGES aggregates confidences and selects the correct answer.

Figure 1: **Examples of CGES vs. SC.** Top: early stopping with high confidence; Bottom: recovering a minority-but-confident answer.

deliberately sample multiple responses per query and then *aggregate* them into a final answer using principled Bayesian inference.

A core component of our framework is *uncertainty estimation*, which has also been studied extensively in generative LLMs. Probability-based methods such as length-normalized scoring [Malinin and Gales, 2021] reduce bias against longer responses, while more recent approaches explicitly account for meaning or learn trainable scoring functions. For example, MARS [Bakman et al., 2024] introduces a semantics-aware weighting of token contributions, and LARS [Yaldiz et al., 2025] formulates uncertainty estimation as a supervised learning problem over token-level scores. Other studies explore Bayesian or distillation-based techniques [Vejendla et al., 2025] for efficient uncertainty estimation. Unlike these works, our aim is not to propose new uncertainty estimation methods. Instead, we adapt and integrate existing techniques–modifying them when appropriate–within the CGES framework.

3 Confidence-Based Approach

In this section, we propose a confidence-based method as an alternative to the majority-vote self-consistency approach. We formalize our setting, introduce a Bayesian framework, provide theoretical guarantees under ideal conditions, and discuss the realistic scenario where confidence scores may be noisy.

3.1 Problem Setting and Notations

Suppose we query a large language model (LLM) multiple times with a given query Q, whose true answer is A, thereby obtaining a sample set $\mathcal{S}$ of responses. Let $\mathcal{U} = \{a_1, a_2, \ldots, a_K\}$ denote the complete set of all possible distinct candidate answers from the LLM, where K is the maximum number of such candidates. We assume the correct answer is within this candidate set and denote its index as $I \in [K]$. Conditioned on (Q, A) and the identity of the correct candidate I, there exists a probabil-

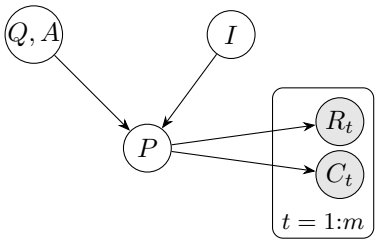


Figure 2: Graphical model for the sampling process.

ity distribution over answers $P = (P_1, \ldots, P_K) \sim f_P(Q, A, I)$, where $P_j = \mathbb{P}[R = a_j \mid Q, A, I]$. Intuitively, this captures the stochastic behavior of the LLM given the query: the likelihood of producing each candidate answer depends both on the query and on which candidate is correct. For each LLM call $t = 1, 2, \ldots$, we draw a response–confidence pair (R_t, C_t) according to $R_t \sim P$ and $C_t \sim f_{C|P}(P)$. That is, the response R_t is drawn i.i.d. from the distribution P, and the confidence signal C_t—a scalar attached to R_t—is a (possibly noisy) proxy derived from P. This structure can be represented as a graphical model, illustrated in Fig. 2.

Idealistic vs. Realistic Assumptions. It is important to distinguish between two types of assumptions. *Idealistic assumptions* are simplifying conditions introduced to design a simple, efficient algorithm and to prove that it achieves optimality under ideal conditions (Theorem 1). These include the assumption of a uniform error distribution and the independence of confidence scores from the true index. In contrast, *realistic assumptions* are those expected to hold in practice and are used both in the derivation of the CGES algorithm and in establishing guarantees under realistic conditions (Theorem 2). These include the i.i.d. sampling assumption and the uniform prior on I. Thus, while all four assumptions 1-2-3-4 hold in the idealized setting, only the weaker pair 1-2—i.i.d. sampling and uniform prior on I—are retained under realistic conditions. The four assumptions are as follows:

1. Given a fixed query Q, with a fixed $\mathcal{U}$ and P, the samples $\{(R_t, C_t)\}_{t=1}^m$ are i.i.d.
2. $I \sim \text{Uniform}(\{1, \ldots, K\})$, reflecting that uncertainty lies in the indexing convention, not in the identity of the correct answer.
3. Under hypothesis $I = i$, the correct answer a_i is emitted with probability C_t, while each incorrect candidate shares the residual probability mass uniformly:
$$\mathbb{P}(R_t = a_i \mid C_t, I = i) = C_t, \qquad \mathbb{P}(R_t = a_j \neq a_i \mid C_t, I = i) = \frac{1 - C_t}{K - 1}.$$
4. The confidence score C_t is independent of the index of the correct answer. Intuitively, the confidence attached to a sample should not depend on which candidate happens to be correct. Formally,
$$\mathbb{P}(C_t \mid I = i) = \mathbb{P}(C_t \mid I = j) \quad \forall i, j \in \{1, \ldots, K\}.$$

Given these assumptions, the objectives of our framework are twofold: (i) to *identify the most probable index $i \in [K]$ corresponding to the true answer*, and (ii) to *quantify the level of confidence in this selection*.

3.2 Bayesian Confidence-Based Framework

We use Bayesian inference to compute posterior probabilities. That is, given an unknown index $I \in \{1, \ldots, K\}$ and a sequence of observed response–confidence pairs, the posterior distribution over I is
$$\mathbb{P}(I = i \mid \text{Obs}) = \frac{\mathbb{P}(\text{Obs} \mid I = i)\, \mathbb{P}(I = i)}{\sum_{k=1}^K \mathbb{P}(\text{Obs} \mid I = k)\, \mathbb{P}(I = k)}.$$
Here, the numerator combines the likelihood of the observations under hypothesis $I = i$ and the prior $\mathbb{P}(I = i)$, while the denominator normalizes across all K competing hypotheses. The set of observations is defined as $\text{Obs} = \{R_1, C_1, R_2, C_2, \ldots, R_m, C_m\}$, where R_t denotes the response at step t and C_t its associated confidence score. This alternating structure reflects how, at each trial, both the raw prediction and its confidence are incorporated.

Since we assume a uniform prior over all hypotheses, i.e., $\mathbb{P}(I = i) = 1/K$, Bayes' rule simplifies to
$$\mathbb{P}(I = i \mid \text{Obs}) = \frac{\prod_t \mathbb{P}(R_t, C_t \mid I = i)}{\sum_{k=1}^K \prod_t \mathbb{P}(R_t, C_t \mid I = k)}.$$
This formulation emphasizes that the posterior is built by multiplying the contributions of each observation and then renormalizing across all indices.

According to Assumption 4, we have
$$\mathbb{P}(C_t \mid I = i) = \mathbb{P}(C_t \mid I = j) \quad \forall i, j \in \{1, \ldots, K\}.$$

Algorithm 1 SCORE: Confidence-Based Bayesian Normalization

Require: Candidate set $\mathcal{U} = \{a_1, \ldots, a_K\}$; samples $\mathcal{S} = \{(R_t, C_t)\}_{t=1}^{m}$ with $R_t \in \mathcal{U}$ and $C_t \in (0, 1)$

for all $a_i \in \mathcal{U}$
  $s_{a_i} \leftarrow \prod_{t:R_t=a_i} C_t \times \prod_{t:R_t \neq a_i} \frac{1-C_t}{K-1}$
return $\text{score}(a_i) = s_{a_i}/Z$ for all a_i

Algorithm 2 Confidence-Guided Early Stopping (CGES)

Require: N questions; threshold γ; calls budget B; scoring routine SCORE

Initialize $\text{scores}_n \leftarrow \text{SCORE}(\{(R_1^n, C_1^n)\})$ for all $n \in [N]$
$D_{\text{rem}} \leftarrow [N]$; calls $\leftarrow N$
for $t = 2$ to B
  $D_{\text{rem}} \leftarrow \{\, n \in D_{\text{rem}} : \max_i \text{scores}_n[i] < \gamma\}$
  if $D_{\text{rem}} = \emptyset$ **break**
  Query LLM for each $n \in D_{\text{rem}}$; calls $+= |D_{\text{rem}}|$
  for all $n \in D_{\text{rem}}$
    $\text{scores}_n \leftarrow \text{SCORE}(\{(R_1^n, C_1^n), \ldots, (R_t^n, C_t^n)\})$
return $\hat{y}_n = a_{\arg\max_i \text{scores}_n[i]}$ $\forall n$, usage$=$ calls$/N$

As a result, the joint distribution factorizes as

$$\mathbb{P}(R_t, C_t \mid I = i) = \mathbb{P}(R_t \mid C_t, I = i)\, \mathbb{P}(C_t),$$

and the marginal $\mathbb{P}(C_t)$ cancels out in the numerator and denominator of Bayes' rule. This leads to the simplified posterior form

$$\mathbb{P}(I = i \mid \text{Obs}) \propto \frac{\prod_t \mathbb{P}(R_t \mid C_t, I = i)}{\sum_{k=1}^{K} \prod_t \mathbb{P}(R_t \mid C_t, I = k)} \quad \triangleq \quad X_i, \tag{1}$$

where X_i represents the posterior mass for hypothesis i.

Finally, in the ideal scenario, we assume that the conditional probability of a response takes the following form:

$$\mathbb{P}(R_t \mid C_t, I = i) = \begin{cases} C_t & \text{if } R_t = a_i, \\ \frac{1-C_t}{K-1} & \text{otherwise.} \end{cases}$$

Intuitively, this means that when the response R_t matches the true answer a_i, it is selected with probability equal to the reported confidence C_t, while the remaining $(K-1)$ incorrect answers share the residual probability mass uniformly.

3.3 Algorithm for Confidence-Based Scoring

Given a set of sampled answers and their confidences $\mathcal{S} = \{(R_t, C_t)\}_{t=1}^{m}$ for a single question, our goal is to convert them into calibrated, comparable probabilities over the unique answer set $\mathcal{U} = \{a_t\}_{t=1}^{K}$. The Bayesian posterior of Eq 1 factorizes into a product of per-sample terms: if the hypothesis (answer) is a_i, then a sample R_t that outputs a_i has likelihood C_t, and any other answer has likelihood $(1 - C_t)/(K - 1)$. We therefore form an unnormalized score s_{a_i} by multiplying these terms across all samples and then normalize across candidates. Algorithm 1 implements this computation.

Algorithm 2 wraps SCORE into an adaptive loop that allocates test-time compute per question. We begin with one sample per question, compute posteriors with SCORE, and maintain the set of unresolved questions D_{rem} whose current top posterior is below a confidence threshold γ. At round $t = 2, \ldots, B$, we query the LLM *only* for $n \in D_{\text{rem}}$, append the new (R_t^n, C_t^n), recompute SCORE on that question's t samples, and remove it from D_{rem} as soon as its top posterior exceeds γ. The process stops when all questions are confident or the budget B is reached, returning the argmax label per question and the average number of LLM calls.

3.4 Theoretical Analysis of Algorithm Performance (Ideal Scenario)

Theorem 1. *Under Assumptions 1, 2, 3, and 4, and provided the confidences are informative i.e., $\mathbb{P}(C_t = 1/K) = 0$, the Bayesian confidence-based aggregator identifies the correct answer with probability tending to one as the number of samples m grows:*

$$\mathbb{P}\big(\arg\max_{i\in[K]} X_i = I\big) \;\longrightarrow\; 1 \qquad \text{as } m \to \infty.$$

In fact, $X_I \to 1$ and $X_k \to 0$ for all $k \neq I$ almost surely.

Proof Sketch. Fix any wrong index $k \neq I$ and compare the (log) likelihood of the observed samples under the hypotheses I vs. k. Each sample contributes a log-likelihood ratio (LLR) increment whose *expected* value is strictly positive whenever the confidence C_t deviates from the uninformative value $1/K$. By the Strong Law of Large Numbers (SLLN), these positive-drift increments accumulate linearly, so the total LLR diverges to $+\infty$. Hence the likelihood under the true index dominates every competitor, forcing the normalized posterior X_I to 1 and all others to 0. A complete formal proof is provided in Appendix A

3.5 Theoretical Analysis of Algorithm Performance (Realistic Scenario)

In contrast to the ideal case, where the data are generated by the same likelihood used by the aggregator, the realistic setting permits model mismatch: the observed answers R_t are drawn from an unknown but fixed (per question) distribution $\mathbf{P} = (P_1, \ldots, P_K)$, while the confidence signal C_t is a noisy proxy produced by an estimator (for example token probabilities). The aggregator itself retains the same one-versus-rest likelihood as in the ideal model; under these conditions, consistency reduces to the sign of the average LLR drift μ_k defined below.

Theorem 2 (Consistency under realistic confidence noise)**.** *Assume 1 and 2. For a fixed question, let $R_t \sim \mathbf{P} = (P_1, \ldots, P_K)$ i.i.d., and $C_t \mid \mathbf{P} \sim f_{C|\mathbf{P}}(\cdot \mid \mathbf{P})$ i.i.d., with $C_t \in (0, 1)$ a.s. and $\mathbb{E}[|\log C_t| + |\log(1 - C_t)|] < \infty$. The aggregator uses the one–versus–rest likelihood from the ideal model. Let $\theta_t = (1 - C_t)/(K - 1)$ and define*

$$\mu_k \;=\; \mathbb{E}_{\mathbf{P},C}\Big[(P_1 - P_k)\,\log\Big(\tfrac{C_t}{\theta_t}\Big)\Big], \qquad k \neq 1.$$

If $\mu_k > 0$ for all $k \neq 1$, then $\mathbb{P}(\arg\max_{i\in[K]} X_i = 1) \to 1$ as $m \to \infty$ and, in fact, $X_1 \to 1$ and $X_k \to 0$ almost surely. If $\mu_{k^\star} < 0$ for some $k^\star \neq 1$, then $X_1 \to 0$ almost surely.

Proof Sketch. As in the ideal case, fix any $k \neq 1$ and consider the log-likelihood ratio (LLR) between hypotheses $I = 1$ and $I = k$. Under the realistic generator, (R_t, C_t) are drawn with $R_t \sim \mathbf{P}$ while the aggregator evaluates likelihoods using C_t and $\theta_t = (1 - C_t)/(K - 1)$. The per-sample LLR increment has conditional mean

$$\mathbb{E}\big[Y_k^{(t)} \mid \mathbf{P}, C_t\big] = (P_1 - P_k)\big(\log C_t - \log \theta_t\big).$$

Averaging over $(\mathbf{P}, C_t)$ gives the drift μ_k. If $\mu_k > 0$, the Strong Law of Large Numbers implies the cumulative LLR grows linearly to $+\infty$, so the likelihood under $I = 1$ dominates and the posterior concentrates on the truth. This condition also covers minority-correct regimes ($P_1 < P_k$) provided C_t is systematically below $1/K$, which flips the sign of the log term and yields positive drift where majority vote would fail. A full proof appears in Appendix B.

4 Experiments

We evaluate CGES on five reasoning benchmarks using two 7B-class models and compare against standard self-consistency (SC) [Wang et al., 2023] and early-stopping self-consistency (ESC) [Li et al., 2024]. *Self-consistency (SC)* aggregates multiple samples by majority vote. *Early-stopping self-consistency (ESC)* halts sampling once predictions within a fixed-size window agree, reducing calls relative to SC. We report *accuracy* of the final answer and *efficiency* as the average number of LLM calls per question. All reported results are averaged over three random seeds to mitigate variance due to stochasticity. Unless noted, decoding and prompting details follow prior work and are provided in Appendix D. Confidence signals C_t are computed using the strategies in Section 4.2.

4.1 Datasets and Models

We evaluate on five benchmarks spanning mathematics and broad knowledge: **AIME24**, consisting of 30 problems from the 2024 American Invitational Mathematics Examination; **MATH500**, a 500-question subset of the MATH benchmark [Hendrycks et al., 2021b] targeting advanced mathematical reasoning; **GSM8K** [Cobbe et al., 2021], 8,500 grade-school math word problems requiring multi-step arithmetic; **MMLU_Pro** [Wang et al., 2024], a more challenging variant of MMLU [Hendrycks et al., 2021a] covering 14 college-level subjects; and **GPQA Diamond** [Rein et al., 2024], expert-written science questions designed to be difficult even for skilled human participants. For the more challenging datasets (AIME24 and GPQA), we employ the stronger *DeepSeek-R1-Distill-Qwen(7B)* [DeepSeek-AI et al., 2025], whereas for the less challenging benchmarks (MATH500, GSM8K, and MMLU_Pro) we use the weaker *Qwen2.5(7B)* [Yang et al., 2025].

4.2 Confidence Estimation Strategies

We compare several strategies for estimating the *scalar* confidence $C_t \in (0, 1)$ of each sampled answer R_t. Our framework operates at the response level, but confidence estimation relies on finer token-level granularities. Specifically, let a response R_t consist of a token sequence $T_1, \ldots, T_L$ with associated autoregressive probabilities $p_1, \ldots, p_L$. Building on this, we now describe three token-based approaches and one verifier-based alternative.

Length-Normalized Scoring (LNS) [Malinin and Gales, 2021]. A natural way to quantify the likelihood of a response is by averaging over token probabilities. The *geometric mean* yields the standard length-normalized score:

$$\mathrm{LNS}_{\mathrm{geom}} \;=\; \exp\Big(\tfrac{1}{L}\sum_{\ell=1}^{L}\log p_\ell\Big)$$

while the *arithmetic mean* provides a simpler length-insensitive proxy, $\mathrm{LNS}_{\mathrm{arith}} = \frac{1}{L}\sum_{\ell=1}^{L} p_\ell$. We set C_t to either of these values and denote them in results as *LNS [Geometric mean]* and *LNS [Arithmetic mean]*.

MARS (Step-Weighted Scoring) [Bakman et al., 2024]. The MARS method generalizes LNS by assigning different weights to different positions in the sequence. Each token T_ℓ receives an exponent

$$w(R, Q, L, \ell) \triangleq \frac{1}{2L} + \frac{u(R, Q, \ell)}{2}$$

so the overall score becomes

$$\bar{P}(R \mid Q, \theta) = \prod_{\ell=1}^{L} p_\ell^{\,w(R,Q,L,\ell)}$$

where θ denotes the parameters of the language model generating token probabilities. Here $u(R, Q, \ell)$ is an importance score for token T_ℓ, such as the semantic change in the output when masking that token. While token-level weighting can be precise, for long reasoning responses most token importance scores become nearly uniform, and computing $u(\cdot)$ for every token is expensive (requiring L calls to a semantic extractor model such as a sentence transformer). To address this, we adopt a *step-wise* variant of MARS: instead of per-token weights, we assign weights at the granularity of reasoning steps or sentence segments. Although the step-importance score is recomputed each iteration, this overhead (6 layers, $\sim$50M parameters) is negligible compared to the 7B-parameter inference model we query for R_t. We denote the resulting confidence as $C_t = \mathrm{MARS}$.

Reward Model Confidence. In addition to the above token-level methods, we also consider a model-based approach that evaluates entire responses directly. A trained reward model assigns a quality score to each R_t, which we use as C_t. In our study, we use `Qwen2.5-Math-PRM-72B` process reward model [Zhang et al., 2025], which outputs a scalar in $(0, 1)$ that correlates with alignment to the ground truth on math-style reasoning. Because it has 72B parameters (far larger than our 7B inference model), using it as the scorer is *impractical* for deployment; we include it as a near-optimal reference to approximate an upper bound on confidence quality, especially on tasks that are in-domain for this PRM. We denote this variant as *RM Confidence*.

Table 1: Accuracy (%) and avg. #Calls across five reasoning tasks; Avg is mean over benchmarks. Parentheses show difference vs. SC (#Calls=16 or SC Acc). For CGES, the first row corresponds to the *Efficient* setting (lower threshold, fewer calls) and the second row to the *Conservative* setting (higher threshold, more calls).[1]

	AIME24		MATH500		GSM8K		GPQA		MMLU_Pro		Avg.	
	#Calls	Acc	#Calls	Acc	#Calls	Acc	#Calls	Acc	#Calls	Acc	#Calls	Acc
SC	16.00	78.89	16.00	82.20	16.00	94.39	16.00	51.01	16.00	61.54	16.00	73.61
ESC (w=4)	11.02	78.89	7.88	82.07	5.22	94.37	11.90	51.18	9.04	61.43	9.01	73.59
	(-4.98)	(+0.00)	(-8.12)	(-0.13)	(-10.78)	(-0.02)	(-4.10)	(+0.17)	(-6.96)	(-0.11)	(-6.99)	(-0.02)
ESC (w=8)	14.40	78.89	11.57	82.20	9.50	94.39	14.87	51.01	12.94	61.54	12.66	73.61
	(-1.60)	(+0.00)	(-4.43)	(+0.00)	(-6.50)	(+0.00)	(-1.13)	(+0.00)	(-3.06)	(+0.00)	(-3.34)	(+0.00)
CGES (Ours)												
LNS[Arithmetic mean]	6.47	78.89	4.69	81.93	4.50	94.26	**2.09**	51.13	6.77	61.56	**4.90**	73.55
	(-9.53)	(+0.00)	(-11.31)	(-0.27)	(-11.50)	(-0.13)	(-13.91)	(+0.12)	(-9.23)	(+0.02)	(-11.10)	(-0.06)
	9.59	78.89	5.81	81.87	4.50	94.26	10.48	51.52	8.29	61.58	7.73	**73.62**
	(-6.41)	(+0.00)	(-10.19)	(-0.33)	(-11.50)	(-0.13)	(-5.52)	(+0.51)	(-7.71)	(+0.04)	(-8.27)	(+0.01)
LNS[Geometric mean]	7.27	78.44	6.64	82.00	5.36	94.36	4.44	51.35	6.78	61.63	6.88	73.56
	(-8.73)	(-0.45)	(-9.36)	(-0.20)	(-10.64)	(-0.03)	(-11.56)	(+0.34)	(-9.22)	(+0.09)	(-9.12)	(-0.05)
	11.56	78.44	6.64	82.00	5.36	94.36	13.26	51.18	10.68	**61.65**	9.90	73.52
	(-4.44)	(-0.45)	(-9.36)	(-0.20)	(-10.64)	(-0.03)	(-2.74)	(+0.17)	(-5.32)	(+0.11)	(-6.10)	(-0.09)
MARS	**5.79**	77.78	6.80	81.93	5.39	**94.42**	3.61	**52.69**	6.68	61.60	5.65	73.28
	(-10.21)	(-1.11)	(-9.20)	(-0.27)	(-10.61)	(+0.03)	(-12.39)	(+1.68)	(-9.32)	(+0.06)	(-10.35)	(-0.33)
	10.83	77.78	6.80	81.93	5.39	94.42	12.14	50.84	10.59	61.53	9.15	73.52
	(-5.17)	(-1.11)	(-9.20)	(-0.27)	(-10.61)	(+0.03)	(-3.86)	(-0.17)	(-5.41)	(-0.01)	(-6.85)	(-0.09)

Table 2: **CGES–PRM (upper bound).** Same conventions as Table 1. Confidences from a large PRM (`Qwen2.5-Math-PRM-72B`; scoring only).

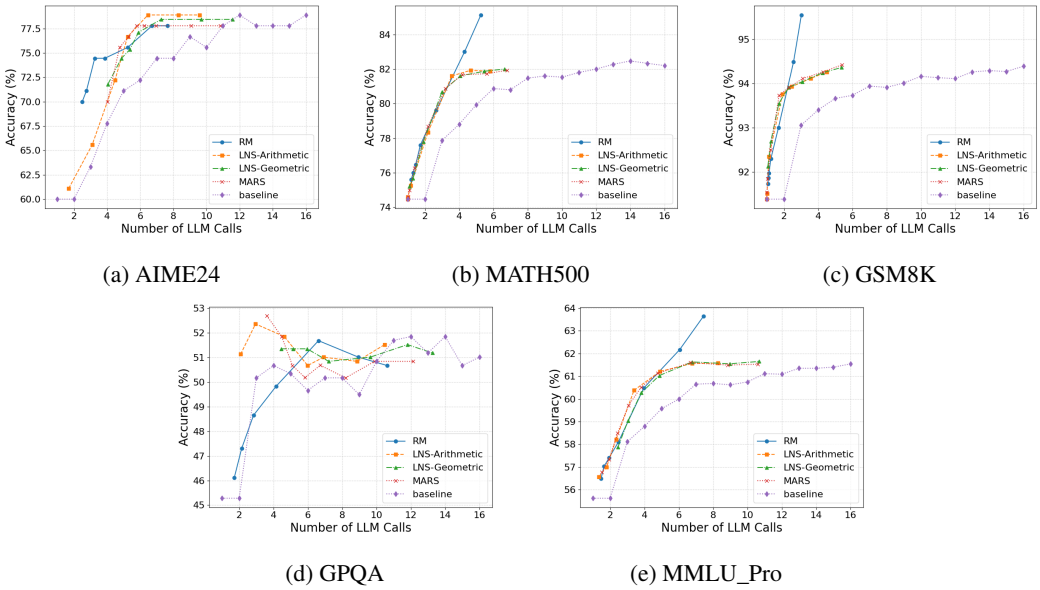
	AIME24		MATH500		GSM8K		GPQA		MMLU_Pro		Avg.	
	#Calls	Acc	#Calls	Acc	#Calls	Acc	#Calls	Acc	#Calls	Acc	#Calls	Acc
RM Confidence	6.71	77.78	4.32	83.00	2.57	94.49	6.62	51.68	6.05	62.17	5.27	73.42
	(-9.29)	(-1.11)	(-11.68)	(+0.80)	(-13.43)	(+0.10)	(-9.38)	(+0.67)	(-9.95)	(+0.63)	(-10.73)	(-0.19)
	7.65	77.78	5.27	85.13	3.02	95.55	10.62	50.67	7.44	63.64	6.40	74.35
	(-8.36)	(-1.11)	(-10.73)	(+2.93)	(-12.98)	(+1.16)	(-5.38)	(-0.34)	(-8.57)	(+2.10)	(-9.60)	(+0.74)

4.3 Results

Table 1 reports accuracy and average number of calls for CGES and baselines with a Self-Consistency budget of $B = 16$. We compare probability-based variants of CGES, which use token-level (LNS) and step-level (MARS) confidence scores.

Across all benchmarks, CGES variants using token-level and step-level confidence (LNS and MARS) significantly reduce the number of model calls compared to both SC and ESC, while maintaining near-identical accuracy. In the configuration achieving the greatest reduction in LLM calls, arithmetic mean LNS achieves an average of just **4.90 calls**—a **69.4%** reduction over SC's fixed budget of 16—while preserving accuracy within -0.06%. Notably, MARS further improves sample efficiency on challenging benchmarks such as GPQA, lowering the average calls from **16.00 to 3.61** with a **+1.68%** accuracy gain. Similarly, on MMLU_Pro, CGES variants reach comparable or better accuracy with fewer than half the calls (e.g., 6.77 calls vs. 16.00 with $+0.02\%$ gain). These results demonstrate that CGES offers a more promising and reliable alternative than local majority-vote stopping rules, enabling adaptive early stopping guided by confidence signals. For MATH500 and GSM8K—the easier tasks—probability-based variants show small accuracy dips despite large call savings. This is expected: when most samples are already correct, marginal improvements hinge on the *calibration* of C_t; noisy probability proxies (LNS/MARS) can be slightly over- or under-confident, limiting accuracy gains. Nevertheless, the efficiency gains are substantial (often $> 2\times$ fewer calls) with accuracy essentially preserved.

[1]The *Efficient* setting corresponds to the smallest γ for which CGES matches or surpasses SC performance, while the *Conservative* setting corresponds to the largest γ considered. If CGES does not reach SC performance, both settings coincide at the largest γ.



(a) AIME24 (b) MATH500 (c) GSM8K

(d) GPQA (e) MMLU_Pro

Figure 3: Accuracy vs. number of LLM calls (B=16) on AIME24 (a), MATH500 (b), GSM8K (c), GPQA (d), and MMLU_Pro (e). CGES achieves near-maximal accuracy with far fewer calls than self-consistency.

Reward-model–based CGES approximates a best-case scenario where confidence estimates are much closer to ground truth than those obtainable from the 7B inference models. Because `Qwen2.5-Math-PRM-72B` is substantially larger and trained for step-level scoring, its signals are stronger than what is practical at inference time. As expected, improvements are most pronounced on math-centric datasets (MATH500, GSM8K), where the PRM's training data overlaps with the task: accuracy rises while calls drop sharply (Table 2). In contrast, on AIME24 and GPQA, the domain mismatch and higher difficulty reduce the utility of PRM scores, yielding small accuracy drops despite fewer calls. Interestingly, on MMLU_Pro, the PRM still supplies useful confidence signals and surpasses SC with substantially fewer calls. Appendix C presents additional results under smaller SC budgets ($B = 4, 8$), showing consistent trends.

Figure 3 shows the accuracy–efficiency trade-off. Each CGES curve is obtained by sweeping the stopping threshold γ; the *baseline* traces self-consistency (SC) at fixed budgets. On AIME24, GSM8K, and MATH500, CGES achieves near-maximal accuracy after only a few calls, whereas SC requires the full budget. **Reward-model (PRM) confidence** converges fastest (often within 3–4 calls), and is shown as a near-ideal reference rather than a deployable setting. Across datasets, curves flatten beyond $\sim$6 calls, indicating diminishing returns and that SC's B=16 is over-provisioned. These results confirm that confidence-guided stopping enables CGES to adaptively terminate sampling early without compromising accuracy.

5 Conclusions

We proposed CGES, a confidence-based Bayesian framework for test-time scaling of LLMs. By treating each response and its confidence as probabilistic evidence, CGES enables early stopping and more reliable aggregation than majority voting. Across five reasoning benchmarks, CGES substantially reduces model calls while maintaining or improving accuracy, outperforming Self-Consistency and early-stopping baselines. Our theoretical analysis further shows correctness under both ideal and noisy confidence assumptions. Overall, confidence-guided aggregation provides a principled solution within the family of test-time scaling methods that sample multiple responses and aggregate them into a single answer. Future work includes (i) developing more accurate confidence estimators to further enhance efficiency and accuracy, and (ii) predicting the required number of samples dynamically from confidence signals.

References

Yavuz Faruk Bakman, Duygu Nur Yaldiz, Baturalp Buyukates, Chenyang Tao, Dimitrios Dimitriadis, and Salman Avestimehr. MARS: Meaning-aware response scoring for uncertainty estimation in generative LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7752–7767, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.419. URL https://aclanthology.org/2024.acl-long.419/.

Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. Forest-of-thought: Scaling test-time compute for enhancing LLM reasoning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=BMJ3pyYxu2.

Yanxi Chen, Xuchen Pan, Yaliang Li, Bolin Ding, and Jingren Zhou. Simple and provable scaling laws for the test-time compute of large language models, 2025. URL https://arxiv.org/abs/2411.19477.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, 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, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, 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, 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, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, 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, Yanhong Xu, 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: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

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 *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021b. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Chengsong Huang, Langlin Huang, and Jiaxin Huang. Divide, reweight, and conquer: A logit arithmetic approach for in-context learning, 2024. URL https://arxiv.org/abs/2410.10074.

Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiaxin Huang. Efficient test-time scaling via self-calibration, 2025. URL https://arxiv.org/abs/2503.00031.

Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. Tree search for language model agents, 2024. URL https://arxiv.org/abs/2407.01476.

Guillaume Lample, Marie-Anne Lachaux, Thibaut Lavril, Xavier Martinet, Amaury Hayat, Gabriel Ebner, Aurélien Rodriguez, and Timothée Lacroix. Hypertree proof search for neural theorem proving, 2022. URL https://arxiv.org/abs/2205.11491.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.291. URL https://aclanthology.org/2023.acl-long.291/.

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 *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ndR8Ytrzhh.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Andrey Malinin and Mark Gales. Uncertainty estimation in autoregressive structured prediction. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=jN5y-zb5Q7m.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling, 2025. URL https://arxiv.org/abs/2501.19393.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

Harshil Vejendla, Haizhou Shi, Yibin Wang, Tunyu Zhang, Huan Zhang, and Hao Wang. Efficient uncertainty estimation via distillation of bayesian large language models, 2025. URL https://arxiv.org/abs/2505.11731.

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 *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=y10DM6R2r3.

Duygu Nur Yaldiz, Yavuz Faruk Bakman, Baturalp Buyukates, Chenyang Tao, Anil Ramakrishna, Dimitrios Dimitriadis, Jieyu Zhao, and Salman Avestimehr. Do not design, learn: A trainable scoring function for uncertainty estimation in generative LLMs. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 691–713, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.41. URL `https://aclanthology.org/2025.findings-naacl.41/`.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL `https://arxiv.org/abs/2412.15115`.

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, 2025. URL `https://arxiv.org/abs/2501.07301`.

A Full Proof of Theorem 1

Proof. Without loss of generality, relabel so that the true index is $I = 1$. Define the unnormalized and normalized posteriors

$$A_j \;:=\; \prod_{t=1}^{m} \mathbb{P}(R_t \mid C_t, I = j), \qquad X_j \;:=\; \frac{A_j}{\sum_{k=1}^{K} A_k}, \qquad j \in [K].$$

To prove the claim, it is enough to show that for every $k \neq 1$, $A_1/A_k \to \infty$ almost surely, which then implies $X_1 \to 1$ and $X_k \to 0$ almost surely.

Step 1: First, for a fixed $k \neq 1$, we define

$$Y_k^{(t)} \;:=\; \log \mathbb{P}(R_t \mid C_t, I = 1) \;-\; \log \mathbb{P}(R_t \mid C_t, I = k).$$

Under Assumption 3, conditioning on C_t we have

$$\log \mathbb{P}(R_t \mid C_t, I = 1) = \begin{cases} \log C_t & \text{if } R_t = a_1, \\ \log\big(\frac{1-C_t}{K-1}\big) & \text{if } R_t \neq a_1, \end{cases}$$

and

$$\log \mathbb{P}(R_t \mid C_t, I = k) = \begin{cases} \log C_t & \text{if } R_t = a_k, \\ \log\big(\frac{1-C_t}{K-1}\big) & \text{if } R_t \neq a_k. \end{cases}$$

Let $\theta_t := \frac{1-C_t}{K-1}$. Using Assumption 1 (i.i.d. across t given P and hence given C_t in this idealized parameterization), the conditional probabilities under the true model $I = 1$ are $\mathbb{P}(R_t = 1 \mid C_t, I = 1) = C_t$ and $\mathbb{P}(R_t = k \mid C_t, I = 1) = \theta_t$.

Step 2: In the next step, by taking the conditional expectation of $Y_k^{(t)}$ given C_t, it follows that

$$\mathbb{E}\Big[Y_k^{(t)} \mid C_t\Big] = \big(C_t - \theta_t\big)\Big(\log C_t - \log \theta_t\Big) = \big(C_t - \theta_t\big) \log\Big(\tfrac{C_t}{\theta_t}\Big).$$

Because $x \mapsto \log x$ is strictly increasing, $(a - b)\log(a/b) > 0$ for $a \neq b$. Here $a = C_t$ and $b = \theta_t$, so the conditional mean is strictly positive whenever $C_t \neq \theta_t$, i.e., whenever $C_t \neq 1/K$. By the informativeness condition $\mathbb{P}(C_t = 1/K) = 0$,

$$\mu_k \;:=\; \mathbb{E}\big[Y_k^{(t)}\big] \;=\; \mathbb{E}\Big[\mathbb{E}\Big[Y_k^{(t)} \mid C_t\Big]\Big] \;>\; 0.$$

Moreover, $|Y_k^{(t)}|$ has finite expectation since $C_t \in (0, 1)$ a.s., making the logs finite.

Step 3: By Assumption 1, $\{Y_k^{(t)}\}_{t=1}^m$ are i.i.d. with $\mathbb{E}[|Y_k^{(t)}|] < \infty$ and $\mathbb{E}[Y_k^{(t)}] = \mu_k > 0$. Finally, the Strong Law of Large Numbers yields

$$\frac{1}{m}\sum_{t=1}^{m} Y_k^{(t)} \xrightarrow{\text{a.s.}} \mu_k \;>\; 0 \quad\Longrightarrow\quad \sum_{t=1}^{m} Y_k^{(t)} \xrightarrow{\text{a.s.}} +\infty.$$

Exponentiating both sides of $\log(A_1/A_k) = \sum_{t=1}^m Y_k^{(t)}$ gives

$$\frac{A_1}{A_k} \;=\; \exp\Big(\sum_{t=1}^{m} Y_k^{(t)}\Big) \xrightarrow{\text{a.s.}} \infty.$$

Since this holds for every $k \neq 1$, we have $A_k/A_1 \to 0$ almost surely for all $k \neq 1$, and therefore

$$X_1 \;=\; \frac{1}{1+\sum_{k\neq 1} A_k/A_1} \xrightarrow{\text{a.s.}} 1, \qquad X_k \;=\; \frac{A_k/A_1}{1+\sum_{j\neq 1} A_j/A_1} \xrightarrow{\text{a.s.}} 0.$$

This completes the proof. □

B Full Proof of Theorem 2

Proof. Without loss of generality, let $I = 1$. For $j \in [K]$, define

$$A_j \;:=\; \prod_{t=1}^{m} \mathbb{P}(R_t \mid C_t, I = j), \qquad X_j \;:=\; \frac{A_j}{\sum_{k=1}^{K} A_k}.$$

It suffices to show that for every $k \neq 1$, $A_1/A_k \xrightarrow{\text{a.s.}} \infty$, which implies $X_1 \to 1$ and $X_k \to 0$ almost surely.

Step 1: In the first step, we fix $k \neq 1$ and set

$$Y_k^{(t)} \;:=\; \log \mathbb{P}(R_t \mid C_t, I = 1) \;-\; \log \mathbb{P}(R_t \mid C_t, I = k).$$

With the model likelihood above and $\theta_t = \frac{1-C_t}{K-1}$,

$$\log \mathbb{P}(R_t \mid C_t, I = 1) = \begin{cases} \log C_t & (R_t = a_1), \\ \log \theta_t & (R_t \neq a_1), \end{cases} \qquad \log \mathbb{P}(R_t \mid C_t, I = k) = \begin{cases} \log C_t & (R_t = a_k), \\ \log \theta_t & (R_t \neq a_k). \end{cases}$$

Step 2: Next by conditioning on $(\mathbf{P}, C_t)$ and using $\mathbb{P}(R_t = a_r \mid \mathbf{P}) = P_r$, we have

$$\mathbb{E}\Big[Y_k^{(t)} \mid \mathbf{P}, C_t\Big] = P_1(\log C_t - \log \theta_t) + P_k(\log \theta_t - \log C_t) = (P_1 - P_k)\big(\log C_t - \log \theta_t\big).$$

Taking expectations over $(\mathbf{P}, C_t)$ gives

$$\mu_k \;:=\; \mathbb{E}\Big[Y_k^{(t)}\Big] = \mathbb{E}_{\mathbf{P},C}\Big[(P_1 - P_k)\log\Big(\tfrac{C_t}{\theta_t}\Big)\Big].$$

By assumption, $\mu_k > 0$ for all $k \neq 1$ and $\mathbb{E}[|Y_k^{(t)}|] < \infty$ (since $C_t \in (0,1)$ a.s. and the logs are integrable).

Step 3: By Assumption 1 and the i.i.d. generation of (R_t, C_t) given $\mathbf{P}$, the increments $\{Y_k^{(t)}\}_{t=1}^m$ are i.i.d. with finite first moment and mean $\mu_k > 0$. The Strong Law of Large Numbers yields

$$\frac{1}{m}\sum_{t=1}^{m} Y_k^{(t)} \to \mu_k \;>\; 0 \text{ almost surely} \quad\Longrightarrow\quad \sum_{t=1}^{m} Y_k^{(t)} \to +\infty \text{ almost surely.}$$

Hence

$$\frac{A_1}{A_k} = \exp\Big(\sum_{t=1}^{m} Y_k^{(t)}\Big) \to \infty \text{ almost surely,}$$

so $A_k/A_1 \to 0$ a.s. for all $k \neq 1$, and therefore $X_1 \to 1$ and $X_k \to 0$ almost surely.

Converse (necessity). If $\mu_{k^\star} < 0$ for some $k^\star \neq 1$, then by the same SLLN argument, $\sum_{t=1}^{m} Y_{k^\star}^{(t)} \to -\infty$ a.s., so $A_1/A_{k^\star} \to 0$ almost surely and thus $X_1 \to 0$ almost surely. (When $\mu_k = 0$ for some k, the LLR has zero drift and the posterior need not concentrate; this is a boundary case.) □

C Additional Results

Additional results are provided in Table 3 and Table 4.

	AIME24		MATH500		GSM8K		GPQA		MMLU_Pro		Avg.	
	#Calls	Acc	#Calls	Acc	#Calls	Acc	#Calls	Acc	#Calls	Acc	#Calls	Acc
SC	4.00	67.78	4.00	78.80	4.00	93.40	4.00	50.67	4.00	58.79	4.00	69.89
CGES (Ours)												
LNS[Arithmetic mean]	3.03 (–0.97)	68.89 (+1.11)	2.59 (–1.41)	79.40 (+0.60)	1.84 (–2.16)	93.48 (+0.08)	1.52 (–2.48)	53.02 (+2.35)	2.76 (–1.24)	59.05 (+0.26)	2.35 (–1.65)	70.77 (+0.88)
	4.00 (0.00)	68.89 (+1.11)	3.75 (–0.25)	79.40 (+0.60)	3.86 (–0.14)	93.48 (+0.08)	4.00 (0.00)	53.02 (+2.35)	3.98 (–0.02)	59.05 (+0.26)	3.92 (–0.08)	70.77 (+0.88)
LNS[Geometric mean]	3.17 (–0.83)	67.99 (+0.21)	2.93 (–1.07)	79.40 (+0.60)	2.14 (–1.86)	93.40 (+0.00)	3.33 (–0.68)	53.54 (+2.87)	3.02 (–0.98)	59.00 (+0.21)	2.92 (–1.08)	70.67 (+0.78)
	4.00 (0.00)	67.78 (+0.00)	3.95 (–0.05)	79.47 (+0.67)	3.99 (–0.01)	93.35 (–0.05)	4.00 (0.00)	53.20 (+2.53)	4.00 (0.00)	59.34 (+0.55)	3.99 (–0.01)	70.63 (+0.74)
MARS	2.48 (–1.52)	67.78 (+0.00)	3.00 (–1.00)	79.00 (+0.20)	2.15 (–1.85)	93.45 (+0.05)	3.00 (–1.00)	53.05 (+2.38)	3.05 (–0.95)	59.06 (+0.27)	2.74 (–1.26)	70.47 (+0.58)
	4.00 (0.00)	72.22 (+4.44)	3.98 (–0.02)	79.07 (+0.27)	3.99 (–0.01)	93.38 (–0.02)	4.00 (0.00)	51.85 (+1.18)	4.00 (0.00)	59.14 (+0.35)	3.99 (–0.01)	71.13 (+1.24)
CGES - Near Ideal Scenario (Ours)												
RM Confidence	1.72 (–2.28)	68.89 (+1.11)	1.80 (–2.20)	78.93 (+0.13)	2.16 (–1.84)	94.16 (+0.76)	3.88 (–0.12)	50.51 (–0.16)	2.41 (–1.59)	58.98 (+0.19)	2.39 (–1.61)	70.29 (+0.40)
	3.68 (–0.32)	73.33 (+5.55)	2.74 (–1.26)	83.27 (+4.47)	2.43 (–1.57)	94.94 (+1.54)	3.97 (–0.03)	50.34 (–0.33)	3.44 (–0.56)	61.97 (+3.18)	3.25 (–0.75)	72.77 (+2.88)

Table 3: Accuracy (%) and avg. #Calls across five reasoning tasks; Avg is mean over benchmarks. Parentheses show difference vs. SC (#Calls=4 or SC Acc). Efficient (first row) vs. Conservative (second row) are two CGES settings.

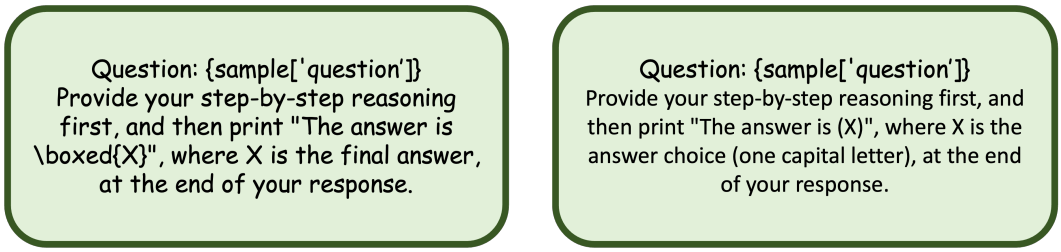
	AIME24		MATH500		GSM8K		GPQA		MMLU_Pro		Avg.	
	#Calls	Acc	#Calls	Acc	#Calls	Acc	#Calls	Acc	#Calls	Acc	#Calls	Acc
SC	8.00	74.45	8.00	81.47	8.00	93.91	8.00	50.17	8.00	60.68	8.00	72.14
CGES (Ours)												
LNS[Arithmetic mean]	4.41 (–3.59)	74.45 (0.00)	4.08 (–3.92)	81.87 (+0.40)	3.41 (–4.59)	93.83 (–0.08)	2.08 (–5.92)	51.12 (+0.95)	4.42 (–3.58)	60.68 (0.00)	3.68 (–4.32)	72.39 (+0.25)
	6.88 (–1.12)	74.45 (0.00)	4.88 (–3.12)	81.87 (+0.40)	4.26 (–3.74)	93.83 (–0.08)	7.56 (–0.44)	51.12 (+0.95)	6.59 (–1.41)	60.68 (0.00)	6.03 (–1.97)	72.39 (+0.25)
LNS[Geometric mean]	4.32 (–3.68)	75.19 (+0.74)	4.67 (–3.33)	81.73 (+0.26)	4.04 (–3.96)	93.78 (–0.13)	4.25 (–3.75)	51.68 (+1.51)	5.78 (–2.22)	60.89 (+0.21)	4.61 (–3.39)	72.65 (+0.51)
	7.71 (–0.29)	72.22 (–2.23)	5.52 (–2.48)	81.80 (+0.33)	5.05 (–2.95)	93.78 (–0.13)	7.99 (–0.01)	51.35 (+1.18)	7.40 (–0.60)	60.95 (+0.27)	6.74 (–1.26)	72.02 (–0.12)
MARS	4.04 (–3.96)	74.45 (0.00)	3.69 (–4.30)	81.60 (+0.13)	5.08 (–2.92)	93.88 (–0.03)	3.56 (–4.44)	52.69 (+2.52)	4.43 (–3.57)	60.75 (+0.07)	4.16 (–3.84)	72.67 (+0.53)
	7.63 (–0.37)	75.56 (+1.11)	5.63 (–2.37)	81.87 (+0.40)	5.08 (–2.92)	93.88 (–0.03)	7.97 (–0.03)	51.68 (+1.51)	7.49 (–0.51)	61.03 (+0.35)	6.76 (–1.24)	72.80 (+0.66)
CGES - Near Ideal Scenario (Ours)												
RM Confidence	1.72 (–6.28)	75.56 (+1.11)	3.26 (–4.74)	82.67 (+1.20)	2.37 (–5.63)	94.44 (+0.53)	5.32 (–2.69)	50.34 (+0.17)	4.42 (–3.58)	61.68 (+1.00)	3.42 (–4.58)	72.94 (+0.80)
	5.70 (–2.30)	75.56 (+1.11)	3.82 (–4.18)	84.47 (+3.00)	2.73 (–5.27)	95.45 (+1.54)	7.20 (–0.80)	50.00 (–0.17)	5.20 (–2.80)	63.19 (+2.51)	4.93 (–3.07)	73.73 (+1.59)

Table 4: Accuracy (%) and avg. #Calls across five reasoning tasks; Avg is mean over benchmarks. Parentheses show difference vs. SC (#Calls=4 or SC Acc). Efficient (first row) vs. Conservative (second row) are two CGES settings.

D Decoding Hyperparameters and Prompt Templates

For CGES, we sweep the stopping threshold γ over the grid $0.70, , 0.75, , 0.80, , 0.85, , 0.90, , 0.95, , 0.99, , 0.999, , 0.9999$ to explore different accuracy–efficiency trade-offs. All experiments, including those for SC, ESC, CGES, and confidence estimation, share the same decoding setup. Specifically, we allow a maximum of 32,768 generation tokens, use a temperature of 0.7, and apply nucleus sampling with top-$p = 1.0$ (i.e., no truncation). Top-k sampling is disabled in all runs.

Prompt Templates. We use dataset-specific answer-format constraints to simplify parsing. Prompt templates are shown in Fig 4.



(a) Prompt used for **MATH500, AIME24, GSM8K**

(b) Prompt used for **GPQA, MMLU_Pro**

Figure 4: Two prompt templates for evaluation.

E The Use of Large Language Models (LLMs)

We used large language models (LLMs) solely to aid with polishing the writing and improving clarity of exposition. No part of the research ideation, methodology, analysis, or experimental results was generated by LLMs. The authors take full responsibility for the content of this paper.