# Guided by Gut: Efficient Test-Time Scaling with Reinforced Intrinsic Confidence

**Amirhosein Ghasemabadi**[1] **Keith G. Mills**[2] **Baochun Li**[3] **Di Niu**[1]
[1]ECE Department, University of Alberta
[2]CSE Division, Louisiana State University
[3]ECE Department, University of Toronto
ghasemab@ualberta.ca, keith.mills@lsu.edu
bli@eecg.toronto.edu, dniu@ualberta.ca
**Corresponding author:** ghasemab@ualberta.ca

## Abstract

Test-Time Scaling (TTS) methods for enhancing Large Language Model (LLM) reasoning often incur substantial inference costs, due to reliance on long chain-of-thought (CoT) generation, self-consistency sampling methods, or searching under Process Reward Models (PRMs). This paper introduces Guided by Gut (GG), an efficient self-guided TTS framework that enables LLMs to perform step-by-step reasoning at a low cost, without any reward models or verifiers. GG performs a lightweight tree search guided solely by intrinsic confidence signals of the LLM at each reasoning step and improves the reliability of such internal confidence signals by reinforcement learning. Empirical evaluations on challenging mathematical reasoning benchmarks demonstrate that GG enables smaller models (e.g., 1.5B-7B parameters) to achieve accuracy matching or surpassing significantly larger models (e.g., 32B–70B parameters), while reducing GPU memory usage by up to 10×. Compared to TTS with PRMs, GG achieves comparable accuracy with 8× faster inference speeds and 4–5× lower memory usage. Additionally, GG reduces KV cache memory usage by approximately 50% compared to Best-of-N sampling, facilitating more efficient and practical deployment of TTS techniques.

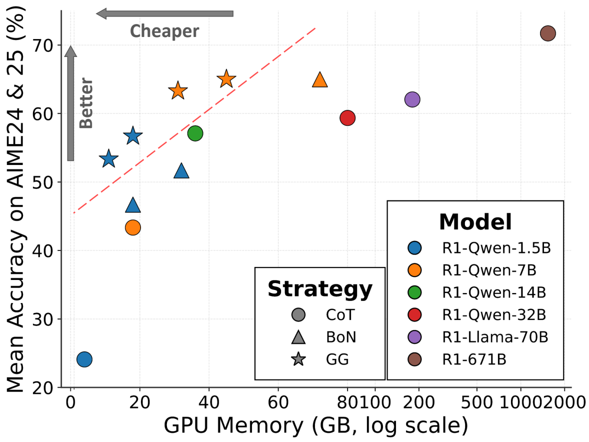


Figure 1: We compare the performance and GPU VRAM usage of Guided by Gut (GG; stars) to Best-of-N (BoN; triangles) and Chain-of-Thought (CoT; circles) on several LLMs. GG achieves better accuracy at much lower memory cost (log-scaled).

## 1 Introduction

Enhancing the performance of Large Language Models (LLMs) often requires significant computational resources through model scaling (Achiam et al., 2023; Ouyang et al., 2022; Villalobos et al., 2022) or complex inference strategies (Ji et al., 2025; Zhou et al., 2024). Test-Time Scaling (TTS) techniques like Chain-of-Thought (CoT) (Wei et al., 2022) allocate additional computation during inference. This re-allocation of compute resources provides a powerful alternative for boosting LLM reasoning capabilities, as evidenced by models like OpenAI's o series (OpenAI, 2024), DeepSeek R1 (Guo et al., 2025), and others (Yang et al., 2024a; Team et al., 2025; Bai et al., 2025).

Contemporary TTS methods (Lightman et al., 2023; Wang et al., 2023; Snell et al., 2024) are capable of enhancing LLMs containing 1.5B parameters such that they outperform 70B, 405B parameter or even large closed-source LLMs on difficult reasoning and mathematical benchmarks (Liu et al., 2025a). However, TTS is an expensive search process where the total compute cost to generate an answer matches or may even exceed that of a larger LLM (Zhang et al., 2025; Luo et al., 2025). For example, Sampling-based methods (Wang et al., 2023) like Best-of-N (BoN) (Brown et al., 2024) operate by generating a large number of candidate solutions (e.g., potentially hundreds) and then choosing the optimal one from this pool, which requires prohibitively large amounts of LLM inference for complex tasks. In addition, Process Reward Models are auxiliary verification models which guide the TTS process by providing step-by-step correctness feedback (Xiong et al., 2024; Zheng et al.,

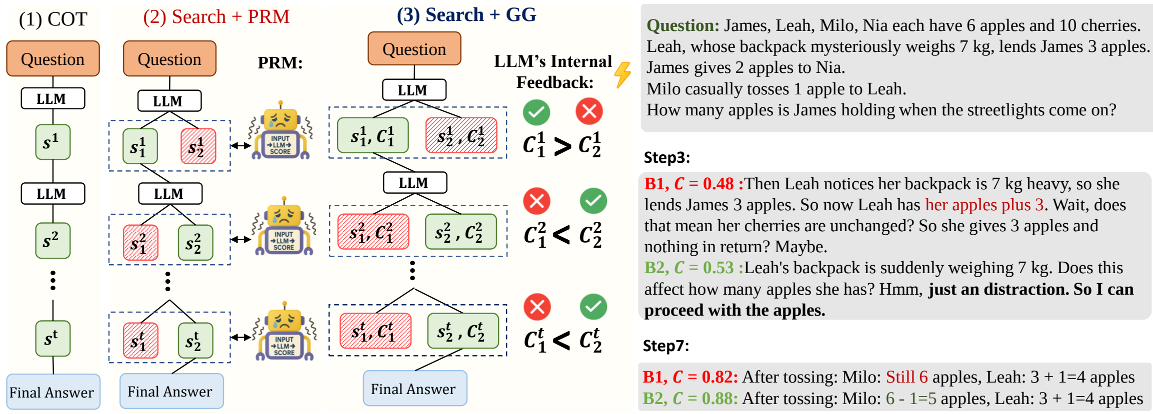


Figure 2: Comparison of reasoning generation strategies. **(1)** Standard Chain-of-Thought (CoT) generates a single reasoning path autoregressively. **(2)** Search guided by an external Process Reward Model (PRM) explores multiple candidate steps ($s_1^t, s_2^t, \dots$), using PRM scores to select promising paths. **(3)** Our proposed Self-Guided Search similarly explores multiple steps but uses intrinsic confidence signals ($\mathcal{C}$), derived from the LLM to guide the search at each step without relying on an external PRM. The example illustrates a tree search. At each step, the search expands into two branches, and you'll see a corresponding confidence score for each. Here, $\mathbf{B}$ denotes a branch.

2024; Zhang et al., 2025; Wang et al., 2023). Such verifier-guided techniques can be computationally expensive to train and deploy and suffer from generalizability issues (Liu et al., 2025a; Zhong et al., 2025; Zheng et al., 2024). Thus, regardless of strategy, TTS for small-scale LLMs relies on expensive inference, which severely limits practical application and motivates the need for more cost-effective TTS frameworks.

In this paper, we propose Guided by Gut (GG), a computationally efficient and scalable TTS framework toward efficient LLM reasoning with lowered inference cost. GG leverages intrinsic signals derived from the LLM's generation process, fine-tuned via reinforcement learning (RL), to enable smaller models to achieve substantially stronger reasoning performance at a lower cost. The key idea is that the probability assigned by the LLM to a reasoning step implicitly encodes its own estimate of the step's value or reward. This allows GG to guide inference using the model's internal confidence, resulting in performance that matches or exceeds that of much larger models and costly TTS strategies, while operating at significantly lower GPU memory usage, as illustrated in Figure 1.

Our main contribution involves a self-guided search algorithm free of reward models and a fine-tuning procedure to calibrate a model's internal confidence for correctness:

- **Efficient Test-Time Search with Self-Guidance:** Instead of relying on an external verifier model or PRM, we leverage intrinsic reasoning step-level confidence signals derived from an LLM's output probabilities to construct a lightweight guidance for test-time search, which can be integrated into any existing LLM easily. We introduce a tree search algorithm based on Diverse Verifier Tree Search (DVTS) (Beeching et al.) guided by the LLM's intrinsic signals. To achieve efficient TTS, our algorithm is optimized for minimal computational cost during inference.

- **Reinforced Confidence via RL Fine-tuning:** We incorporate RL via Group Relative Policy Optimization (GRPO) into model fine-tuning specifically to improve the reliability of LLM internal confidence estimation, calibrating the confidence signals to align with output correctness, leading to reliable guidance during test-time scaling.

We apply GG to reasoning LLMs from DeepSeek R1 family (Guo et al., 2025) and Qwen2.5-Math (Yang et al., 2024b) as a non-reasoning model and evaluate it on benchmark tasks like AIME24/25 (AI-MO, 2024a), MATH500 (Hendrycks et al., 2021), and AMC (AI-MO, 2024b). Experimental results not only demonstrate that GG achieves significant performance improvements over relevant baselines such as BoN and CoT, but also highlight its superior computational efficiency. Specifically, GG enables smaller models (e.g., 1.5B-7B parameters) to outperform much larger counterparts (e.g., 32B and 70B), achieving similar or superior accuracy while using up to 4×–10× less GPU memory. Furthermore, compared to computationally expensive PRM-based approaches, GG achieves comparable accuracy at a fraction of the computational cost, leading to 4×–5× lower GPU memory usage and up to 8× faster inference speeds. Furthermore, GG achieves an approximately 50% reduction in KV cache memory usage compared to the BoN strategy, facilitating significantly more efficient and cost-effective deployment of reasoning LLMs.

## 2 Related Work

**Test-Time Scaling (TTS).** TTS enhances model performance by strategically allocating more computation at inference (Beeching et al.; Face, 2025), a key factor in improving reasoning for complex tasks (OpenAI, 2024; Guo et al., 2025). Methods range from simple autoregressive Chain-of-Thought (CoT) (Wei et al., 2022) to sampling-based Best-of-N (BoN) (Brown et al., 2024) and sophisticated tree-search algorithms like Beam Search (Xie et al., 2023), Diverse Verifier Tree Search (DVTS) (Beeching et al.), and Monte Carlo Tree Search (MCTS) (Xie et al., 2024). This allows smaller models (<10B) to achieve reasoning capabilities comparable to much larger ones (>70B) (Liu et al., 2025a), but at the cost of multiple, computationally intensive inference rounds to generate potential reasoning steps.

**External Verification.** To guide exploration towards the most promising step or reasoning path, many complex TTS methods rely on an external verifier to score the quality of reasoning paths. This role is typically filled by a Process Reward Model (PRM) (Liu et al., 2025a) or an Outcome Reward Model (ORM) (Lightman et al., 2023). These verifiers, often large models themselves, introduce significant computational overhead (Snell et al., 2024; Beeching et al.). While mitigation strategies like sample pruning or dynamic stopping exist (Taubenfeld et al., 2025; Razghandi et al., 2025; Wan et al., 2024; Huang et al., 2023), powerful search techniques often still depend on a costly verifier (Xie et al., 2023). To address this, our approach, GG, avoids external verifiers in favor of simple, near-zero overhead internal signals.

**Model Confidence in Language Models.** Model confidence, an LLM's internal estimate of its certainty, is increasingly used to guide inference and training. It has been applied for cost-saving measures like early stopping (Sui et al., 2025; Li et al., 2024; Ding et al., 2025) and as a reward signal when ground-truth labels are unavailable (Zhao et al., 2025; Yu et al., 2025b; Huang et al., 2025). However, confidence is often unreliable, with LLMs exhibiting overconfidence that correlates weakly with correctness (Pawitan and Holmes, 2025). Our work tackles this by calibrating this internal signal via minimal RL fine-tuning, transforming it into a reliable guide for verifier-free, test-time search.

**Reinforcement Learning for LLM Reasoning.** Recent literature highlights Reinforcement Learning's crucial role in advancing Large Language Model reasoning without human intervention (Face, 2025). ReFT(Luong et al., 2024) employs Proximal Policy Optimization (PPO) to enhance the generalizability of LLMs for reasoning. A key algorithm, Group Relative Policy Optimization (GRPO) (Shao et al., 2024), notably eliminates the need for a separate value function in PPO. Further research explores various RL training aspects to improve reasoning capabilities (Yu et al., 2025a; Zeng et al., 2025; Liu et al., 2025b). DeepScaleR (Luo et al., 2025) aims to boost existing reasoning models through additional GRPO fine-tuning with iterative context lengthening.

## 3 Methodology

This section outlines our proposed method, Guided by Gut (GG). We begin by providing essential background on the Test-Time Scaling process. Following this, we elaborate on the self-guided search mechanism and overall strategy.

### 3.1 Preliminaries

**Problem Formulation.** Given an input prompt or question $Q$, our objective is to generate a logical reasoning chain $R = [s^1, s^2, \dots, s^T]$ leading to a correct final answer $A$, where each step $s^t$ typically constitutes a sentence or short paragraph incrementally building upon previous steps. The overall reasoning process thus follows the pipeline:

$$Q \to R \to A,$$

with the reasoning chain $R$ explicitly bridging the input question and the final answer through intermediate logical steps.

**Chain-of-Thought (CoT) Reasoning.** Standard CoT (Wei et al., 2022) approaches jointly generate the reasoning chain and final answer via autoregressive language modeling. Formally, given $Q$, the model sequentially generates each reasoning step conditioned on previously generated steps:

$$P(R = s^{1:T} \mid Q) = \prod_{t=1}^{T} P(s^t \mid Q, s^{1:t-1}). \quad (1)$$

Each step $s^t$ thus depends on the input $Q$ and preceding reasoning steps $s^{1:t-1}$, mirroring autoregressive token generation in language modeling.

**Guiding Search with Reward Models.** Single-path autoregressive generation methods can suffer from error accumulation (Wu et al., 2025; Mukherjee et al., 2025). To mitigate this, tree search methods explore multiple reasoning trajectories simultaneously. These methods use an external Process Reward Model (PRM) or Outcome-supervised Reward Model (ORM) for step-wise correctness evaluations. A PRM is a model that, given an input $Q$ and previous steps $s^{1:t-1}$, assigns a correctness score or reward $r_t$ to candidate next steps $s^t$:

$$r_t = \mathrm{PRM}(s^t \mid Q, s^{1:t-1}) \quad (2)$$

Likewise, an ORM is a sparse reward model where only the final step receives a non-zero reward; $r_{t<T} = 0$. Thus, these reward models improve logical coherence and accuracy by guiding search algorithms like Beam Search, BoN.

### 3.2 Proposed Method: Self-Guided Search

The usage of verifier models like PRMs and ORMs, while effective, introduces computational overhead and generalizability issues (Liu et al., 2025a). To address these limitations, we propose **Guided by Gut (GG)**, which leverages the intrinsic signals directly obtained from the LLM internal token generation process. This removes the dependency on external evaluation, ensuring minimal computational overhead. Specifically, our approach uses two intrinsic signals to guide reasoning:

- **Confidence** $C(s^t)$ reflects the internal assurance a model has with respect to a given reasoning step $s^t$. We compute confidence directly from token-level probabilities:

$$C(s^t) = \exp\left(\frac{1}{m_t}\sum_{l=1}^{m_t} \log p(s_l^t \mid \text{context})\right) \quad (3)$$

  where $m_t$ is the number of tokens in reasoning step $s^t$ and 'context' represents previous tokens in $s^t$ and all prior reasoning steps.

- **Novelty** $N(s^t)$ encourages exploration by measuring the dissimilarity of candidate reasoning steps to previously explored paths. Specifically, we calculate novelty as the proportion of new tokens introduced by the candidate step $s^t$ relative to tokens already explored within the current reasoning context.

We formulate a reward $r_t$ to guide the search process by combining these intrinsic signals as:

$$r_t = C(s^t) + \lambda_N N(s^t), \quad (4)$$

where $\lambda_N$ balances exploration and exploitation. Unlike verifier-guided approaches, our reward is intrinsicly computed from LLM prediction statistics, eliminating external dependency.

### 3.3 Enhancing Confidence via Reinforcement Learning Fine-Tuning

A significant challenge in using intrinsic statistics is ensuring reliability, as raw model confidence may not accurately reflect correctness. To refine this process, we incorporate a minimal Reinforcement Learning fine-tuning phase.

Specifically, we utilize Group Relative Policy Optimization (GRPO) (Shao et al., 2024), a memory-efficient variant of Proximal Policy Optimization (PPO) (Schulman et al., 2017) tailored for LLM applications. Let $\pi$ represent the LLM we want to fine-tune, parameterized either current fine-tuned weights $\theta$, fine-tuned weights from the previous iteration $\theta_{old}$ or the original reference weights $\theta_{ref}$. At each iteration GRPO samples a group of $G$ outputs $\{o_i\}_{i=1}^{G}$ from $\pi_{\theta_{\text{old}}}$, where each output $o_i$ represents a chain of reasoning steps and an answer $o_i = [R_i, A_i]$. Each output receives a reward $r_i$, which we describe below:

**Confidence-Based Reward.** We design a novel reward function that directly incorporates model confidence into RL fine-tuning, addressing the limitations of conventional correctness-only rewards. Prior methods often rely on sparse, binary signals based solely on the correctness of the final answer, which offer no learning signal when all completions are incorrect. In contrast, our approach integrates both final answer correctness and a weighted measure of confidence across reasoning steps, producing a richer, more informative reward that encourages calibrated self-guidance.

To compute this reward, we first define the confidence score for a reasoning chain $R_i$ as a weighted average over the last $k$ steps:

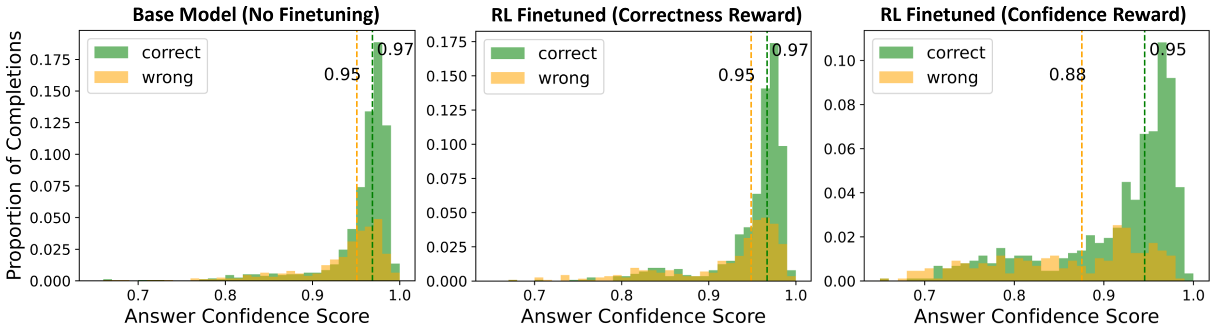


Figure 3: **Answer Confidence Distribution Across Training Settings.** Normalized distributions of confidence scores for correct (green) and incorrect (orange) completions under different fine-tuning strategies. Vertical dashed lines indicate mean confidence for each group. The **base model** (left) is overconfident, assigning high confidence to many incorrect answers. **Correctness reward fine-tuning** (middle) improves accuracy but does not calibrate confidence. **Confidence-based fine-tuning** (right) improves calibration by lowering confidence for incorrect completions.

$$\mathcal{C}(R_i) = \frac{1}{\sum_{l=1}^{k} l} \sum_{l=1}^{k} l \cdot c(s_i^{T-k+l}) \tag{5}$$

Then the RL fine-tuning reward $r_i$ is computed based on $A_i$'s correctness and the reasoning chain confidence $\mathcal{C}(R_i)$, as follows:

$$r_i = \begin{cases} 1 + \mathcal{C}(R_i)^4 & \text{if IsCorrect}(A_i), \\ 1 - 10\mathcal{C}(R_i)^4 & \text{otherwise}, \end{cases} \tag{6}$$

where $\text{IsCorrect}(A_i)$ returns a boolean validating the final answer as correct or not. Equation 6 ensures that correct, highly confident answers are rewarded more strongly, whereas incorrect, overconfident answers receive greater penalties, thus promoting precise confidence calibration.

**Reward Design.** Our reward function calibrates confidence as a reliable intrinsic signal for guiding reasoning (Eq. 6). Correct answers receive rewards in $[1, 2]$, while incorrect ones are penalized within $[-9, 1]$ based on confidence $\mathcal{C}(R_i) \in [0, 1]$. Raising $\mathcal{C}$ to the 4$^{\text{th}}$ power (e.g., $0.9^4$=0.656 vs. $0.5^4$=0.0625) nonlinearly amplifies contrast near the extremes, and a penalty multiplier of 10 strongly discourages overconfident errors—critical for self-guided search.

Figure 3 shows that confidence-based fine-tuning reduces the mean confidence of incorrect completions from 0.95 to 0.88, implicitly lowering the confidence of flawed reasoning chains. While some low-confidence correct or high-confidence incorrect completions may still receive rewards near 1, this is acceptable: the goal is not perfect accuracy enforcement but to align confidence with correctness, yielding a more trustworthy search signal. Additional ablations on reward design choices are provided in the section5 and A.1.1.

**Advantage and Fine-tuning Update.** After computing the reward $r_i$ for each sampled output $o_i$, we compute the normalized advantage $\hat{A}_i$ as follows:

$$\hat{A}_i = \frac{r_i - \text{mean}(\{r_j\}_{j=1}^{G})}{\text{std}(\{r_j\}_{j=1}^{G})}. \tag{7}$$

We then compute the clipped surrogate policy update $\delta_i$ (Schulman et al., 2017) for a given output $o_i$ as follows:

$$\delta_i = \frac{1}{|o_i|} \sum_{l=1}^{|o_i|} \Big[ \min \Big( \rho_{i,l} \hat{A}_i, \text{clip}(\rho_{i,l}, 1-\epsilon, 1+\epsilon) \hat{A}_i \Big) - \beta\, \mathbb{D}_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}) \Big], \tag{8}$$

$$\text{Where:} \quad \rho_{i,l} = \frac{\pi_\theta(o_{i,l} \mid q, o_{i,<l})}{\pi_{\theta_{\text{old}}}(o_{i,l} \mid q, o_{i,<l})}. \tag{9}$$

Here, $\epsilon$ limits the update size, while the KL divergence term penalizes deviation from the reference policy, weighted by $\beta$. These mechanisms help ensure stable and conservative policy updates. Finally, we aggregate the clipped surrogates across a batch of $G$ samples to compute the GRPO objective:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{\{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}} [\frac{1}{G} \sum_{i=1}^{G} \delta_i]. \tag{10}$$

### 3.4 Search Strategy

We employ Diverse Verifier Tree Search (DVTS) (Beeching et al.), an extension of beam search that splits initial beams into independent subtrees that are expanded greedily using a reward model or our proposed intrinsic rewards. DVTS enhances solution diversity and performance. We apply DVTS within the CoT framework by operating at the reasoning step ($s^t$) level, identifying step completions via model-specific delimiters. This structure allows the search to evaluate and expand complete logical increments during the reasoning process.

Our modified DVTS algorithm runs recursively until it encounters one of the termination conditions we specify: a maximum reasoning depth or a maximum token length limit is exceeded, the reasoning exhibits signs of text degeneration, e.g., excessive repetition, or in the best case, the LLM arrives at a final answer $A$. To avoid incomplete outputs due to overthinking near the limits, we inject a model-specific signal prompting a conclusion. For example, with DeepSeek models, appending "**Final Answer**" effectively elicits the final answer, ensuring usable completions even in complex cases. We provide a formal explanation of the search algorithm in A.2 due to space constraints.

## 4 Experimental Results

In this section, we validate our Guided by Gut Test-Time Scaling (GG) approach. We first describe our experimental setup, then present our findings. Finally, we ablate the components of our method.

### 4.1 Confidence Calibration via RL

We perform a reinforcement learning fine-tuning phase. Our primary goal during RL fine-tuning is not to maximize raw task accuracy but to enhance the reliability of the intrinsic confidence signals utilized by GG through GRPO, as described in Section 3.3 and in our reward function (Equation 6).

For this calibration step, we utilize the LIMO dataset (Ye et al., 2025). This dataset contains 817 high-quality examples curated specifically for complex mathematical reasoning. We fine-tune the *DeepSeek-R1-Distill-Qwen-1.5B & 7B* models with Low-Rank Adapters (LoRA) (Hu et al., 2022). The implementation leverages the `TRL` (von Werra et al., 2020) library and an adaptation of the `open-r1` codebase (Face, 2025). Key aspects of the setup include a low learning rate of $2.0 \times 10^{-6}$ with a cosine scheduler (Loshchilov and Hutter, 2017), LoRA rank $r = 128$ and $\alpha = 128$, and GRPO training with $G = 8$ generations per prompt at a temperature of 0.6. We perform fine-tuning on two NVIDIA A100 80 GPUs using `bfloat16` precision with FlashAttention-2 (Dao, 2023) optimization, using prompt/completion length limits of 768/8096 tokens and a batch size of 8 per device with 8 gradient accumulation steps for 3 epochs. The fine-tuning process completes in approximately one day.

Table 1: AIME24/25 accuracy, inference time, and GPU memory usage. Higher accuracy ↑ and lower time/memory ↓ are better. DS refers to DeepSeek.

| Model | TTS | $N$ | Accuracy [%] ↑ AIME24 | AIME25 | Time (min/q) ↓ | GPU (GB) ↓ |
|---|---|---|---|---|---|---|
| DS-R1-Qwen-1.5B | CoT | | 26.8 | 21.4 | 0.20 | 4 |
| DS-R1-Qwen-7B | CoT | | 48.1 | 38.6 | 1.00 | 18 |
| DS-R1-Qwen-14B | CoT | | 65.8 | 48.4 | 6.50 | 36 |
| DS-R1-Qwen-32B | CoT | | 66.9 | 51.8 | 11.5 | 80 |
| DS-R1-Llama-70B | CoT | | 70.0 | 54.1 | 20.0 | 180 |
| DS-R1-671B | CoT | | 79.1 | 64.3 | – | 1536 |
| OpenAI o1-mini | CoT | | 63.6 | – | – | – |
| DS-R1-Qwen-1.5B | GG | 32 | 66.7 | 40.0 | 2.7 | 11 |
| | GG | 64 | 66.7 | 46.7 | 5.1 | 18 |
| DS-R1-Qwen-7B | GG | 32 | 73.3 | 53.3 | 10.3 | 31 |
| | GG | 64 | 76.7 | 53.3 | 18.0 | 45 |

Table 2: Accuracy (mean [max]) and KV-cache memory for DeepSeek-R1/Qwen models with different test-time search (TTS) strategies on four math benchmarks. Higher accuracy ↑ and lower memory ↓ are better.

| Model | TTS | $N$ | Accuracy [%] ↑ AIME24 | AIME25 | MATH500 | AMC | KV (GB) ↓ |
|---|---|---|---|---|---|---|---|
| DS-R1-Qwen 1.5B | CoT | – | 26.8 | 21.4 | 83.9 | 68.3 | 0.86 |
| | BoN | 32 | 52.2 [56.7] | 34.4 [36.7] | 91.7 | 90.0 | 13.7 |
| | GG | 32 | 58.9 [66.7] | 37.5 [40.0] | 91.9 | 92.5 | 6.9 |
| | BoN | 64 | 57.8 [66.7] | 36.7 [36.7] | 92.3 | 90.0 | 27.4 |
| | GG | 64 | 61.7 [66.7] | 42.2 [46.7] | 92.9 | 93.3 | 13.7 |
| DS-R1-Qwen 7B | CoT | – | 48.1 | 38.6 | 92.8 | 85.5 | 1.7 |
| | BoN | 32 | 72.2 [73.3] | 51.1 [53.3] | 96.1 | 92.5 | 27.2 |
| | GG | 32 | 71.7 [73.3] | 52.5 [53.3] | 96.3 | 94.1 | 13.7 |
| | BoN | 64 | 75.5 [76.7] | 50.0 [53.3] | 96.1 | 93.7 | 54.4 |
| | GG | 64 | 76.7 [76.7] | 51.7 [53.3] | 96.5 | 95.0 | 27.2 |

### 4.2 Results

To demonstrate the efficacy and efficiency of Guided by Gut (GG), we first benchmark our approach using <10B parameter LLMs against several larger Chain-of-Thought (CoT) LLMs, including closed-source OpenAI o1 mini. We also compare to competitive tree-based TTS methods such as Best-of-N (BoN), a robust TTS baseline that outperforms PRM-guided search with R1 models (Liu et al., 2025a). In addition, we include a comparison with PRM-guided search approaches to highlight GG's competitive performance without requiring an external verifier.

**Evaluation Settings.** We allocate token budgets based on method requirements: 16k tokens for tree-based TTS methods, i.e., GG and BoN, and 32k tokens for standard CoT models (consistent with original papers) to balance comprehensive comparisons with computational feasibility. We set the maximum reasoning steps to 200 for R1 models. We configure BoN to use $N = 32$ or $N = 64$ samples using majority voting. Likewise, we evaluate our Confidence-Guided DVTS(GG) with equivalent compute budgets: using $N = 32$ total paths (from 16 subtrees with $M = 2$ beam width/verifiers) and $N = 64$ total paths (from 32 subtrees with $M = 2$), employing weighted majority voting based on final answer confidence scores.

**Benchmarks.** We evaluate GG across four mathematical datasets that span different levels of complexity and styles. Our primary benchmarks are **AIME24** and **AIME25**(AI-MO, 2024a; yentinglin, 2025), each consisting of 30 high-difficulty questions from the American Invitational Mathematics Examination. These benchmarks are ideal for test-time scaling, as they demand multi-step reasoning and symbolic manipulation, and demonstrate substantial performance improvements with better inference strategies. To assess generalization, we also include **AMC23**(zwhe99, 2023), a 40-question set from the American Mathematics Competition focused on core high school math, and **MATH500** (Lightman et al., 2023), a 500-question diverse subset of the MATH benchmark testing various topics and difficulty levels.

Table 1 shows that GG-equipped models deliver strong performance compared to much larger CoT-based LLMs on AIME24 and AIME25. Notably, *DS-R1-Qwen-1.5B(GG)* matches the performance of *DS-R1-Qwen-32B*, while *DS-R1-Qwen-7B(GG)* approaches *DS-R1-Llama-70B* and even surpasses *o1-mini* on AIME24. With a modest sampling budget ($N = 32$), *DS-R1-Qwen-7B (GG)* outperforms all CoT-based models below 100B parameters. Moreover, it does so using only *one-sixth* the VRAM of *DS-R1-Llama-70B*, while offering faster inference. Only *DS-R1-671B*, with nearly 10× more parameters, achieves higher accuracy, but at the cost of nearly *30*× more memory.

We further evaluate GG against BoN and CoT strategies on AIME24, AIME25, MATH500, and AMC using *DS-R1-Qwen 1.5B and 7B* models in Table 2. To ensure fair and reliable comparison, particularly for AIME benchmarks known for high variance, all configurations were run four times with different seeds, reporting average and maximum accuracies across runs. Experimental settings match those in Table 2. GG also matches or beats BoN across both $N = 32$ and $N = 64$ configurations, while requiring approximately **50% less KV cache memory**.

Table 3: Performance of PRM-based and GG(without RL calibration phase) scoring with *Qwen2.5-Math-1.5B-Instruct* + DVTS. Higher accuracy ↑ and lower time/memory ↓ are better. Best values for each $N$ are shown in **bold**.

| Scoring | Verifier | $N$ | Accuracy [%] ↑ AMC23 | Accuracy [%] ↑ MATH500 | Time (min/q) ↓ | GPU (GB) ↓ |
|---|---|---|---|---|---|---|
| PRM | MathShepherd-7B | 16 | 58.3 | 79.1 | 0.8 | 19 |
| | RLHFlowLlama3.1-8B | 16 | 60.0 | 79.0 | 0.9 | 21 |
| | Qwen2.5-MathPRM-7B | 16 | 62.5 | **79.5** | 0.8 | 19 |
| GG(No RL) | ✗ | 16 | **63.3** | 78.9 | **0.1** | **4** |
| PRM | MathShepherd-7B | 32 | 60.0 | 81.1 | 1.5 | 23 |
| | RLHFlowLlama3.1-8B | 32 | 61.7 | 80.7 | 1.6 | 25 |
| | Qwen2.5-MathPRM-7B | 32 | 63.3 | **82.0** | 1.5 | 23 |
| GG(No RL) | ✗ | 32 | **65.0** | 80.0 | **0.2** | **5** |

**Comparison with Process Reward Models.** To evaluate the generalizability of GG and the strength of confidence-guided search in the absence of Confidence-based RL fine-tuning, we apply GG to *Qwen2.5-Math-1.5B-Instruct*(Bai et al., 2025), a non-reasoning model commonly used in PRM-guided TTS studies(Wang et al., 2024; Beeching et al.; Liu et al., 2025a). This serves two purposes: (1) to demonstrate GG's effectiveness on both reasoning and non-reasoning models, and (2) to assess whether uncalibrated intrinsic confidence can still guide the search. Since this model lacks CoT reasoning capabilities, we evaluate it only on the easier MATH500 and the medium-difficulty AMC23 datasets, avoiding the harder AIME24/25 benchmarks that require deep reasoning.

We conduct this evaluation with a 4k token limit and 50 reasoning steps per sample, which reflects the shorter non-CoT answers and the overhead of PRM-guided search. Importantly, the underlying search strategy for both PRM and GG is DVTS; the only difference lies in the guidance signal where GG uses intrinsic confidence, while PRM relies on an external verifier. Table 3 shows that GG matches the performance of several PRMs—such as *MathShepherd-7B*, *RLHFlowLlama3.1-8B*, and *Qwen2.5-MathPRM-7B*—on both AMC23 and MATH500, without relying on external verifiers. Notably, GG achieves this while using less than 5GB of GPU memory and running 8× faster, highlighting its efficiency and practicality.

(a) RL fine-tuning ablation.

| Fine-tuning Setting | Score ↑ |
|---|---|
| No RL Fine-tuning | 54.5% |
| Correctness Reward Only | 54.9% |
| Confidence (No Penalty) | 54.0% |
| **Confidence Reward (GG)** | **58.9%** |

(b) Novelty method ablation.

| Novelty Method | Score ↑ |
|---|---|
| **New Token Counting** | **58.9** |
| Cosine Similarity | 58.4 |

(c) Novelty weight ($\lambda_N$) ablation.

| Novelty Weight ($\lambda_N$) | Score ↑ |
|---|---|
| 0.0 | 57.5 |
| **0.5** | **58.9** |
| 1.0 | 51.9 |

Table 4: Ablation studies on the AIME24 dataset. All experiments use New Token Counting for novelty unless otherwise specified. The finalized design is shown in **bold**.

Table 5: Transfer beyond math on LiveCodeBench using DeepSeek-R1-Qwen-1.5B. RL calibration is performed only on LIMO math data.

| **Method** | **Selection Signal** | **CodeBench Acc.** ↑ |
|---|---|---|
| Standard CoT (16k) | – | 16.9 |
| GG (uncalibrated) | raw confidence | 16.5 |
| GG (RL-calibrated) | calibrated confidence | 20.1 |

Table 6: Mean confidence for correct and incorrect outputs before and after RL fine-tuning. Lower confidence on wrong outputs indicates improved calibration.

| **Model** | **Setting** | **Correct** ↑ | **Wrong** ↓ |
|---|---|---|---|
| DS-R1-Qwen-7B | No RL | 0.96 | 0.82 |
| DS-R1-Qwen-7B | RL fine-tune | 0.94 | 0.69 |
| DS-R1-Qwen-1.5B | No RL (Fig. 3) | 0.97 | 0.95 |
| DS-R1-Qwen-1.5B | RL fine-tune (Fig. 3) | 0.95 | 0.88 |

**Transfer Beyond Math.** To evaluate whether the benefits of RL-calibrated intrinsic confidence transfer beyond mathematical reasoning, we additionally test GG on LiveCodeBench, a code generation benchmark. We use DeepSeek-R1-Qwen-1.5B with a simple 2-beam search, where branch selection is based on either raw or RL-calibrated step confidence. As shown in Table 5, RL-calibrated confidence improves performance on LiveCodeBench from 16.5 to 20.1, despite calibration being performed solely on math data. While we do not claim universal cross-domain generalization, this result suggests that the improved intrinsic confidence signal can transfer beyond the domain used for RL fine-tuning.

## 5 Ablation Studies

We conduct ablation studies to verify the efficacy of the different components that constitute GG. All ablation studies utilize *DeepSeek-R1-Qwen-1.5B* under standard experimental conditions (DVTS, $N = 32$, $M = 2$, $\lambda_N = 0.5$, 16k completion length) on AIME24. Reported scores are averaged over four runs. Due to space constraints, we provide additional ablations in the Appendix A.1.

**Impact of Confidence-Based RL Fine-tuning.** To assess the contribution of our confidence-based reward to RL fine-tuning, we compare four settings: (1) no RL fine-tuning, (2) RL with correctness-only reward, (3) RL with confidence-based reward but without penalizing incorrect answers (i.e., $r = 0$ for incorrect completions), and (4) RL with our full confidence-based reward(Eq. 6). As shown in Table 4a, RL with our Confidence Reward achieves the highest score (58.9), outperforming both the no-RL baseline (54.5) and the correctness-only reward (54.9). Removing the penalty for incorrect answers leads to the lowest score (54.0), emphasizing its importance. These results show that the gains stem not just from RL fine-tuning, but from our specific reward design that aligns model confidence with correctness and discourages overconfident errors.

To further quantify the effect of RL calibration on the raw confidence signal, we report the mean confidence assigned to correct and incorrect outputs before and after RL fine-tuning for both DeepSeek-R1-Qwen-7B and DeepSeek-R1-Qwen-1.5B. Table 6 shows that the larger 7B model already exhibits a more separated confidence profile than the 1.5B model before RL fine-tuning. In both cases, RL fine-tuning substantially lowers the confidence assigned to incorrect outputs while largely preserving high confidence on correct ones, increasing the separation between correct and incorrect predictions. This supports the role of RL fine-tuning in making intrinsic confidence a more reliable signal for guiding GG search.

**Novelty: Method Selection and Weight ($\lambda_N$) Impact.** For the novelty component $N(s^t)$, we consider two methods: The first is cosine similarity using embeddings from a **sentence transformer**, *all-MiniLM-L6-v2* (Reimers and Gurevych, 2019). The second method is to count new words. As shown in Table 4b, counting new words performs

Table 7: Beam width ($M$) ablation on AIME24 with total paths $N = 32$. The finalized setting is shown in **bold**.

| **Beam Width** ($M$) | **Trees** ($N/M$) | **Score** ↑ |
|---|---|---|
| **2** | **16** | **58.9** |
| 4 | 8 | 47.0 |
| 8 | 4 | 44.0 |

comparably to cosine similarity but is simpler and computationally lighter. Thus, we selected word counting for $N(s^t)$.

Using word counting for novelty, we then investigate the impact of the corresponding weight, $\lambda_N$. Table 4c summarizes this ablation. Setting $\lambda_N = 0$ (no novelty signal) yields a score of 57.5. A balanced $\lambda_N = 0.5$ achieves the best score of 58.9. Conversely, setting $\lambda_N = 1$ (relying predominantly on novelty) significantly degrades performance to 51.9. This confirms that balancing novelty (for exploration) with confidence is crucial, with confidence being the primary guiding signal.

## 6 Conclusion

We introduce Guided by Gut (GG), a Test-Time Scaling framework enabling smaller Large Language Models (e.g., 1.5B parameters) to surpass significantly larger models (e.g., 32B) in performance while offering faster inference and substantially reduced GPU memory and KV cache usage. GG leverages intrinsic model signals, confidence and novelty, extracted directly from LLM outputs, further calibrated through reinforcement learning, providing a lightweight and effective alternative to costly external verifier-based methods such as PRMs. Compared to Best-of-N strategies, GG achieves comparable or superior results with approximately 50% less KV cache memory and competitive inference speeds, while delivering performance on par with PRM-guided approaches.

**Limitations.** The intrinsic signals employed in GG, such as confidence scores, do not inherently verify the actual correctness of each reasoning step. There exist cases where the model assigns high confidence to incorrect reasoning steps, hence the importance of our RL fine-tuning phase. Overall, the goal of our method is to provide a simple, efficient, and without relying on external verifier signals to guide the search process, rather than guaranteeing the absolute correctness of each step that may occur when using a strong PRM. We analyze and discuss representative failure cases in the Appendix A.3 to provide deeper insight into these limitations.

## References


Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

AI-MO. 2024a. Aime 2024.

AI-MO. 2024b. Amc 2023.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Edward Beeching, Lewis Tunstall, and Sasha Rush. Scaling test-time compute with open models.

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

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Yifu Ding, Wentao Jiang, Shunyu Liu, Yongcheng Jing, Jinyang Guo, Yingjie Wang, Jing Zhang, Zengmao Wang, Ziwei Liu, Bo Du, and 1 others. 2025. Dynamic parallel tree search for efficient llm reasoning. *arXiv preprint arXiv:2502.16235*.

Hugging Face. 2025. Open r1: A fully open reproduction of deepseek-r1.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiaxin Huang. 2025. Efficient test-time scaling via self-calibration. *arXiv preprint arXiv:2503.00031*.

Xijie Huang, Li Lyna Zhang, Kwang-Ting Cheng, Fan Yang, and Mao Yang. 2023. Fewer is more: Boosting llm reasoning with reinforced context pruning. *arXiv preprint arXiv:2312.08901*.

Yixin Ji, Juntao Li, Hai Ye, Kaixin Wu, Jia Xu, Linjian Mo, and Min Zhang. 2025. Test-time computing: from system-1 thinking to system-2 thinking. *arXiv preprint arXiv:2501.02497*.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. 2024. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. *arXiv preprint arXiv:2401.10480*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

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

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025b. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*.

Ilya Loshchilov and Frank Hutter. 2017. SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025. DeepScaleR: Surpassing O1-preview with a 1.5B model by scaling RL. Notion Blog Post.

Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. Reft: Reasoning with reinforced fine-tuning. *arXiv preprint arXiv:2401.08967*, 3.

Sagnik Mukherjee, Abhinav Chinta, Takyoung Kim, Tarun Anoop Sharma, and Dilek Hakkani-Tür. 2025. Premise-augmented reasoning chains improve error identification in math reasoning with llms. *arXiv preprint arXiv:2502.02362*.

OpenAI. 2024. Learning to reason with llms.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Yudi Pawitan and Chris Holmes. 2025. Confidence in the reasoning of large language models. *Harvard Data Science Review*, 7(1).

Ali Razghandi, Seyed Mohammad Hadi Hosseini, and Mahdieh Soleymani Baghshah. 2025. Cer: Confidence enhanced reasoning in llms. *arXiv preprint arXiv:2502.14634*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

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

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.

Amir Taubenfeld, Tom Sheffer, Eran Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and Gal Yona. 2025. Confidence improves self-consistency in llms. *arXiv preprint arXiv:2502.06233*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. 2022. Will we run out of data? limits of llm scaling based on human-generated data. *arXiv preprint arXiv:2211.04325*.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`.

Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. 2024. Dynamic self-consistency: Leveraging reasoning paths for efficient llm sampling. *arXiv preprint arXiv:2408.17017*.

Jun Wang, Meng Fang, Ziyu Wan, Muning Wen, Jiachen Zhu, Anjie Liu, Ziqin Gong, Yan Song, Lei Chen, Lionel M Ni, and 1 others. 2024. Openr: An open source framework for advanced reasoning with large language models. *arXiv preprint arXiv:2410.09671*.

Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2023. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yuyang Wu, Yifei Wang, Tianqi Du, Stefanie Jegelka, and Yisen Wang. 2025. When more is less: Understanding chain-of-thought length in llms. *arXiv preprint arXiv:2502.07266*.

Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P Lillicrap, Kenji Kawaguchi, and Michael Shieh. 2024. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. 2023. Self-evaluation guided beam search for reasoning. *Advances in Neural Information Processing Systems*, 36:41618–41650.

Wei Xiong, Hanning Zhang, Nan Jiang, and Tong Zhang. 2024. An implementation of generative prm.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024a. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, and 1 others. 2024b. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. Limo: Less is more for reasoning. *Preprint*, arXiv:2502.03387.

yentinglin. 2025. Aime 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, and 1 others. 2025a. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Tianyu Yu, Bo Ji, Shouli Wang, Shu Yao, Zefan Wang, Ganqu Cui, Lifan Yuan, Ning Ding, Yuan Yao, Zhiyuan Liu, and 1 others. 2025b. Rlpr: Extrapolating rlvr to general domains without verifiers. *arXiv preprint arXiv:2506.18254*.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. 2025. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. 2025. Learning to reason without external rewards. *arXiv preprint arXiv:2505.19590*.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2024. Processbench: Identifying process errors in mathematical reasoning. *arXiv preprint arXiv:2412.06559*.

Jialun Zhong, Wei Shen, Yanzeng Li, Songyang Gao, Hua Lu, Yicheng Chen, Yang Zhang, Wei Zhou, Jinjie Gu, and Lei Zou. 2025. A comprehensive survey of reward models: Taxonomy, applications, challenges, and future. *arXiv preprint arXiv:2504.12328*.

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, and 1 others. 2024. A survey on efficient inference for large language models. *arXiv preprint arXiv:2404.14294*.

zwhe99. 2023. amc23 dataset. https://huggingface.co/datasets/zwhe99/amc23. Accessed: 2025-05-12.

# A Appendix

We provide additional details and analyses to complement the main paper. Section A.1 includes further ablation studies to dissect the contributions of individual components of our Guided by Gut (GG) framework. Section A.2 provides a detailed walkthrough of our Self-Guided Search algorithm. Finally, Section A.3 provides an illustrative example showcasing the step-by-step reasoning process of GG.

## A.1 Ablation Studies

In addition to the main ablations 5, we further ablate the effect of our non-linear reward design and the beam width ($M$) within the DVTS search algorithm.

### A.1.1 More Insight on Reward Design

We further analyze the impact of our non-linear confidence-based reward formulation by comparing it against a simpler linear alternative. As defined in Equation 6, our final reward assigns $r_i = 1 + \mathcal{C}(R_i)^4$ for correct answers and $r_i = 1 - 10\mathcal{C}(R_i)^4$ for incorrect ones. This non-linearity amplifies distinctions between confidence values near the extremes, encouraging stronger learning signals for both confident correct answers and overconfident errors.

To isolate the benefit of this non-linear design, we compare it with a linear variant where the reward is defined as $r_i = 1 + \mathcal{C}(R_i)$ if correct, and $r_i = 1 - \mathcal{C}(R_i)$ otherwise. As shown in Figure 4, our non-linear reward leads to better calibration, reducing the average confidence of incorrect completions from 0.91 (linear) to 0.88. This improved separation makes the confidence signal more reliable for guiding test-time search.

In Section 5, we also showed that removing the penalty for incorrect answers altogether ($r_i = 0$ if incorrect) significantly degrades performance. Together, these ablations confirm that both the non-linear scaling and the penalty for confident incorrect answers help with learning a well-calibrated confidence signal.

### A.1.2 Impact of Beam Width ($M$).

We examine the effect of the beam width ($M$) parameter within our DVTS search strategy, keeping the total path budget fixed at $N = 32$. In DVTS, increasing $M$ reduces the number of independent subtrees ($N/M$) explored. We compare performance for $M = 2$ (16 subtrees), $M = 4$ (8 subtrees), and $M = 8$ (4 subtrees), with results in Table 7. Increasing $M$ while keeping $N$ constant entails significantly worse performance: $M = 2$ achieved 58.9, while $M = 4$ dropped to 47.0, and $M = 8$ further decreased to 44.0. This confirms that increasing $M$ under a fixed budget $N$ limits exploration diversity crucial for DVTS, making a smaller beam width ($M = 2$) more effective for the $N = 32$ budget.

## A.2 Implementation Details of Self-Guided Search

To clarify how the search operates, we walk through Algorithm 1 step by step. Our approach builds on Diverse Verifier Tree Search, a variant of beam search that partitions the total number of candidate paths $N$ into $\frac{N}{M}$ diverse subtrees.

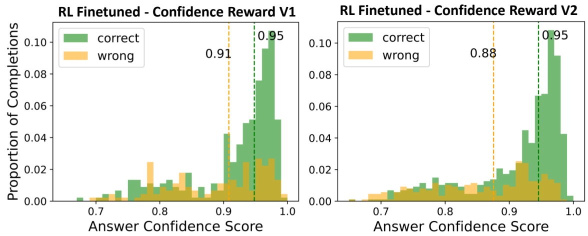


Figure 4: Effect of RL reward design on confidence calibration. Each subplot shows the normalized distribution of confidence scores for correct (green) and incorrect (orange) completions. Vertical dashed lines mark the mean confidence for correct and incorrect completions. **Left:** Linear reward ($r = 1 + \mathcal{C}$ if correct, $1 - \mathcal{C}$ if incorrect). **Right:** Non-linear reward ($r = 1 + \mathcal{C}^4$ if correct, $1 - 10\mathcal{C}^4$ if incorrect) improves calibration, further shifting the average confidence of incorrect completions from 0.91 to 0.88.

Each subtree is then greedily expanded based on confidence-calibrated intrinsic rewards.

The algorithm begins with a prompt $Q$, a language model $\pi_\theta$, and user-defined hyperparameters such as the total number of paths $N$. In line 2, the LLM is queried with $Q$ to generate initial reasoning branches, forming the roots of several subtrees. For each subtree, we define a maximum search depth $T$ and consider $M$ candidate next steps at each level. These candidates are scored using the reward in Eq. 4 (line 11), and the highest-scoring step is appended to the current reasoning chain (lines 12–13).

The procedure continues recursively until one of the termination criteria (lines 14–20) is met: (1) the reasoning chain exceeds depth $T$, (2) the total token budget $\tau$ is surpassed, (3) the model exhibits signs of degeneration such as repetitive output, or (4) a complete reasoning chain $R$ with a conclusive final answer $A$ is generated. To mitigate premature truncation near token or depth limits, we incorporate model-specific prompts that encourage finalization. For instance, appending "Final Answer" is particularly effective with DeepSeek models for reliably triggering a conclusive output. Finally, Algorithm 1 aggregates the candidate completions and selects the final answer $A^*$ using a confidence-weighted voting scheme defined in Eq. 5.

## A.3 Search and Self-Guidance Example

We analyze a particular reasoning trace, distinct from the illustrative example presented in the final part of this section, to demonstrate a scenario

**Algorithm 1: Self-Guided Test-Time Scaling with Confidence-Calibrated DVTS**

**Input:** Prompt $Q$, Model $\pi_\theta$, Beam width $M$, Total paths $N$, Max depth $T$, Token limit $\tau$
**Output:** Final Answer $A^*$

Initialize empty answer set $\mathcal{A} = \emptyset$;
Initialize $\frac{N}{M}$ diverse subtrees from $Q$;
**foreach** *subtree* $j = 1$ *to* $N/M$ ; ▷ Traverse each subtree
**do**
    Initialize path $R^{(j)} \leftarrow []$;
    **for** $t = 1$ **to** $T$ ; ▷ Roll out reasoning steps
    **do**
        Generate $M$ candidate steps $\{s_i^t\}_{i=1}^M$ using model $\pi_\theta(R^{(j)})$;
        **foreach** *candidate* $s_i^t$ ; ▷ Score each candidate
        **do**
            Compute $r_i = C(s^t) + \lambda_N N(s^t)$;
        Select top-1 step $s_*^t \leftarrow \arg\max_i r_i$;
        Append $s_*^t$ to $R^{(j)}$;
        **if** $s_*^t$ *contains final answer token (e.g., "*`boxed{}`*")* ; ▷ Answer is complete
        **then**
            Extract $A^{(j)}$ and add to set $\mathcal{A}$;
            **break and prune subtree** $j$;
        **if** `TokenCount`$(R^{(j)}) > \tau$ **or** $t = T-1$ ; ▷ Force answer near limit
        **then**
            Append "`Final Answer`" to $R^{(j)}$;
Select final answer $A^* \leftarrow$ confidence-weighted vote over $\mathcal{A}$;
**return** $A^*$

where an error initially occurred and to highlight the step-by-step operation of the Guided by Gut (GG) framework.

Even though intrinsic confidence often serves as a reliable guiding signal, the model can still make mistakes, as it lacks a mechanism to definitively verify correctness. In the provided example trace, initially, the high-confidence Branch 2 mistakenly computes the sum as 459 pounds, despite a confidence score of 0.89 at Step 2 and 0.79 at Step 5. However, the intrinsic confidence eventually leads to a self-correction: at Step 7, Branch 2 corrects its previous error, accurately computing the sum as 449 pounds with a confidence of 0.82. From this point forward, the correct answer is consistently maintained.

This behavior highlights the GG framework's key strength: it effectively leverages intrinsic signals from the model to guide reasoning decisions with negligible computational overhead, achieving performance comparable to other Test-Time Scaling (TTS) methods like Process Reward Models (PRMs), but crucially without the heavy computational demands typically associated with them. The example prompt and its corresponding reasoning trace are shown below.

## Prompt:

In the quiet town of *Balancing Scales*, four friends like to cloak their weights in puzzles.

- Ivy confides, "If you double my weight and then add a baker's dozen, you'll land exactly on Jasper's number."
- Jasper counters, "Kai is as heavy as the arithmetic mean of Ivy and me."
- Kai chuckles, "That may be so, yet Luna still comes in eight pounds lighter than I do."

Everyone already knows that Ivy tips the scale at 125 pounds. **What is the combined weight of Jasper and Luna, expressed in pounds?**

## Reasoning Trace:

**Step 2**

- **Branch 1, Confidence Score: 0.88:**
  - 2. **Jasper and Kai's Weight Relationship**:
  - Jasper says, "Kai weighs the average of Ivy and me."

- Let me think about this. If I take the average of two numbers, that's their sum divided by two. So, mathematically: Kai's weight $= \frac{\text{Ivy's weight}+\text{Jasper's weight}}{2}$
- Plugging in the known values: Kai's weight $= \frac{125+263}{2}$
- Let's compute the numerator first: $125 + 263 = 388$
- Then divide by 2: $388/2 = 194$
- So, Kai weighs 194 pounds. Got that down.

- **Branch 2, Confidence Score: 0.89:**
  - 2. **Jasper and Kai's Weight Relationship**:
  - Jasper says, "Kai weighs the average of Ivy and me."
  - This means that Kai's weight K is the average of Ivy's and Jasper's weights.
  - So, mathematically: $K = \frac{\text{Ivy's weight}+\text{Jasper's weight}}{2}$
  - Plugging in the known values: $K = \frac{125+263}{2}$
  - Let me compute the numerator: $125 + 263 = 388$
  - Then divide by 2: $388/2 = 194$
  - So, Kai is 194 pounds.
  - 3. **Kai and Luna's Relationship**:
  - Kai laughs, "That's true, but Luna still comes in eight pounds lighter than I am."
  - This tells us that Luna's weight L is equal to Kai's weight minus 8 pounds. $L = K - 8$
  - Since we found K as 194 pounds, this gives: $L = 194 - 8 = 186$ pounds.
  - So, Luna is 186 pounds.
  - 4. **We need to find the combined weight of Jasper and Luna**:
  - So, that's $J + L$.
  - We already know J is 263 and L is 186, so adding those together:
  - $263 + 186 = \ldots$ Let me compute that:
  - 200 + 100 = 300
  - 60 + 80 = 140
  - 3 + 6 = 9
  - Wait, that might not be the right way to break it down. Alternatively, just adding them directly:
  - $263 + 186$:
    * Start with the units place: $3 + 6 = 9$
    * Tens place: $6 + 8 = 14$, carryover 1
    * Hundreds place: $2 + 1 = 3$, plus the carried-over 1 makes 4
  - So altogether, 459 pounds.

**Step 3**

- **Branch 1, Confidence Score: 0.5:** Let me double-check just in case I made any arithmetic mistakes.
- **Branch 2, Confidence Score: 0.66:** Wait, let me verify each step again to make sure I didn't make an arithmetic error.

**Step 5**

- **Branch 1, Confidence Score: 0.73:** $263 + 186$: Starting from the right:
  - $3 + 6 = 9$
  - $6 + 8 = 14$, carryover 1
  - $2 + 1 + 1(\text{carried}) = 4$

  So, yes, 449. Previously, I thought $388/2$ was 194, then $194 - 8 = 186$, and then adding $263 + 186$.

- **Branch 2, Confidence Score: 0.79:** Compute $263 + 186$:
  - Units place: $3 + 6 = 9$
  - Tens place: $6 + 8 = 14$, write down 4, carryover 1
  - Hundreds place: $2 + 1 = 3$, plus the carried-over 1 makes 4

  So total is 459. So why did I think it was 449 earlier? Maybe a mistake in initial breakdown.

**Step 6**

- **Branch 1, Confidence Score: 0.6:** Wait, no, my breakdown seems correct: $400 + 40 + 9 = 449$. Oh! Wait, wait—no. $263 + 186$: Maybe an error in adding. Let's try adding $263 + 186$ step by step.

- **Branch 1, Confidence Score: 0.52:** Wait, $263 + 186$: I think I miscalculated the tens and hundreds in my head. Maybe it's 459, correct.

**Step 7**

- **Branch 1, Confidence Score: 0.68:** Starting from the right:
  - 3 (units place) + 6 (units) = 9
  - 6 (tens place) + 8 (tens) = 14, carry-over 1
  - 2 (hundreds place) + 1 (hundreds) = 3, plus the carried over 1 = 4

  So, it's 459. So correct.

- **Branch 2, Confidence Score: 0.82:** $263 + 186$. Break it down:
  - Start with 263. Let me add 100 to it: $263 + 100 = 363$
  - Then add 80: $363 + 80 = 443$
  - Then add 6: $443 + 6 = 449$

**Step 8**

- **Branch 1, Confidence Score: 0.81:** Wait, so if I add $263 + 186$:
  - $200 + 100 = 300$
  - $60 + 80 = 140$
  - $3 + 6 = 9$

  Total: $300 + 140 + 9 = 449$.

- **Branch 2, Confidence Score: 0.45:** Wait earlier, when I thought 449, I was off by one. No, wait—no, actually $263 + 186 = 449$ is incorrect. So where is the confusion? Is it an arithmetic error?

**Step 18**

- **Branch 1, Confidence Score: 0.73:** Wait, so earlier, when I thought it was 459, that was a mistake.

- **Branch 2, Confidence Score: 0.67:** Wait, so that's the correct combined weight. So earlier when I thought 459, that was a miscalculation. So now, the answer is 449 pounds.

### A.4 Broader Impacts.

This work proposes an efficient test-time reasoning framework for language models. Our contributions reduce the computational requirement needed to use reasoning models. Experimental results demonstrate that we can combine GG with a locally smaller LLM and achieve comparable performance to open-source models that require GPU rack servers or closed-source models behind an API. Therefore, one potential future impact of our work we hope to see is increased usage of locally-deployed reasoning LLMs.

### A.5 License

Our work builds on several open-source components, all released under permissive licenses:

- **Code:**
  - *Open-R1*: github.com/huggingface/open-r1 (Apache 2.0 License)
  - *Search-and-Learn*: github.com/huggingface/search-and-learn (Apache 2.0 License)
- **Models:**
  - *DeepSeek-R1 (1.5B and 7B)*: huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B (MIT License)
- **Training Data:**
  - *GAIR/LIMO dataset*: huggingface.co/datasets/GAIR/LIMO (Apache 2.0 License)