

Can 1B LLM Surpass 405B LLM? Rethinking Compute-Optimal Test-Time Scaling

Runze Liu^{1,2,*}, Junqi Gao^{1,3}, Jian Zhao⁴, Kaiyan Zhang², Xiu Li², Biqing Qi^{1,†}, Wanli Ouyang¹ and Bowen Zhou^{1,2,†}

¹Shanghai AI Laboratory, ²Tsinghua University, ³Harbin Institute of Technology, ⁴BUPT

Test-Time Scaling (TTS) is an important method for improving the performance of Large Language Models (LLMs) by using additional computation during the inference phase. However, current studies do not systematically analyze how policy models, Process Reward Models (PRMs), and problem difficulty influence TTS. This lack of analysis limits the understanding and practical use of TTS methods. In this paper, we focus on two core questions: **(1)** What is the optimal approach to scale test-time computation across different policy models, PRMs, and problem difficulty levels? **(2)** To what extent can extended computation improve the performance of LLMs on complex tasks, and can smaller language models outperform larger ones through this approach? Through comprehensive experiments on MATH-500 and challenging AIME24 tasks, we have the following observations: **(1)** The compute-optimal TTS strategy is highly dependent on the choice of policy model, PRM, and problem difficulty. **(2)** With our compute-optimal TTS strategy, extremely small policy models can outperform larger models. For example, a **1B** LLM can exceed a **405B** LLM on MATH-500. Moreover, on both MATH-500 and AIME24, a **0.5B** LLM outperforms **GPT-4o**, a **3B** LLM surpasses a **405B** LLM, and a **7B** LLM beats **o1** and **DeepSeek-R1**, while with **higher** inference efficiency. These findings show the significance of adapting TTS strategies to the specific characteristics of each task and model and indicate that TTS is a promising approach for enhancing the reasoning abilities of LLMs. Our website is available at <https://ryanliu112.github.io/compute-optimal-tts>.

1. Introduction

Large Language Models (LLMs) have shown significant improvements across a variety of domains (OpenAI, 2023; Hurst et al., 2024; Anthropic, 2023; OpenAI, 2024; DeepSeek-AI et al., 2025). Recently, OpenAI o1 (OpenAI, 2024) has demonstrated that Test-Time Scaling (TTS) can enhance the reasoning capabilities of LLMs by allocating additional computation at inference time, making it an effective approach for improving LLM performance (Qwen Team, 2024; Kimi Team et al., 2025; DeepSeek-AI et al., 2025).

TTS approaches can be divided into two main categories: (1) Internal TTS, which trains the LLMs to “think” slowly with long Chain-of-Thought (CoT) (OpenAI, 2024; DeepSeek-AI et al., 2025), and (2) External TTS, which improves the reasoning performance via sampling or search-based methods with fixed LLMs (Wu et al., 2024; Snell et al., 2024). The key challenge of external TTS is how to

* Work done during an internship at Shanghai AI Laboratory

† Corresponding authors: Biqing Qi (qibiqing@pjlab.org.cn), Bowen Zhou (zhoubowen@tsinghua.edu.cn)

Can 1B LLM Surpass 405B LLM? Rethinking Compute-Optimal Test-Time Scaling

Runze Liu^{1,2,*}, Junqi Gao^{1,3}, Jian Zhao⁴, Kaiyan Zhang², Xiu Li², Biqing Qi^{1,†}, Wanli Ouyang¹ and Bowen Zhou^{1,2,†}

¹Shanghai AI Laboratory, ²Tsinghua University, ³Harbin Institute of Technology, ⁴BUPT

*警告：该PDF由GPT-Academic开源项目调用大语言模型+Latex翻译插件一键生成，版权归原文作者所有。翻译内容可靠性无保障，请仔细鉴别并以原文为准。项目Github地址 https://github.com/binary-husky/gpt_academic/。项目在线体验地址 <https://auth.gpt-academic.top/>。当前大语言模型: Qwen2.5-72B-Instruct, 当前语言模型温度设定: 0.3。为了防止大语言模型的意外谬误产生扩散影响，禁止移除或修改此警告。

测试时扩展（Test-Time Scaling, TTS）是一种通过在推理阶段使用额外计算来提高大型语言模型（LLMs）性能的重要方法。然而，当前的研究并未系统地分析策略模型、过程奖励模型（Process Reward Models, PRMs）和问题难度如何影响TTS。这种分析的缺乏限制了对TTS方法的理解和实际应用。在本文中，我们关注两个核心问题：**(1)** 在不同的策略模型、PRMs和问题难度水平下，最优的测试时计算扩展方法是什么？**(2)** 扩展计算在多大程度上可以提高LLMs在复杂任务上的性能，较小的语言模型是否可以通过这种方法超越较大的模型？通过在MATH-500和具有挑战性的AIME24任务上的全面实验，我们有以下观察：**(1)** 计算最优的TTS策略高度依赖于策略模型、PRM和问题难度的选择。**(2)** 通过我们的计算最优TTS策略，极小的策略模型可以超越较大的模型。例如，一个**1B**的LLM可以在MATH-500上超过**405B**的LLM。此外，在MATH-500和AIME24上，**0.5B**的LLM优于**GPT-4o**，**3B**的LLM超越了**405B**的LLM，而**7B**的LLM则胜过了**o1**和**DeepSeek-R1**，同时具有**更高的推理效率**。这些发现表明，将TTS策略适应于每个任务和模型的具体特征的重要性，并表明TTS是提高LLMs推理能力的一种有前景的方法。我们的网站可在<https://ryanliu112.github.io/compute-optimal-tts>访问。

1. Introduction

大型语言模型（LLMs）在多个领域中展示了显著的改进 (OpenAI, 2023; Hurst et al., 2024; Anthropic, 2023; OpenAI, 2024; DeepSeek-AI et al., 2025)。最近，OpenAI o1 (OpenAI, 2024) 展示了测试时扩展（TTS）可以通过在推理时分配额外的计算来增强LLMs的推理能力，使其成为提高LLM性能的有效方法 (Qwen Team, 2024; Kimi Team et al., 2025; DeepSeek-AI et al., 2025)。

TTS方法可以分为两大类：(1) 内部TTS，通过训练LLMs进行长时间的链式思考（CoT）来“慢慢思考” (OpenAI, 2024; DeepSeek-AI et al., 2025)，和 (2) 外部TTS，通过采样或基于搜索的方法来提高固定LLMs的推理性能 (Wu et al., 2024; Snell et al., 2024)。外部TTS的关键挑战是如何最优地扩展计算，即为每个问题分配最优的计算资源 (Snell et al., 2024)。当前的TTS方法通过过程奖励模型（PRMs）来指导生成过程并选择最终答案，从而有效地扩展测试时的计算 (Wu et al., 2024; Snell et

* Work done during an internship at Shanghai AI Laboratory

† Corresponding authors: Biqing Qi (qibiqing@pjlab.org.cn), Bowen Zhou (zhoubowen@tsinghua.edu.cn)

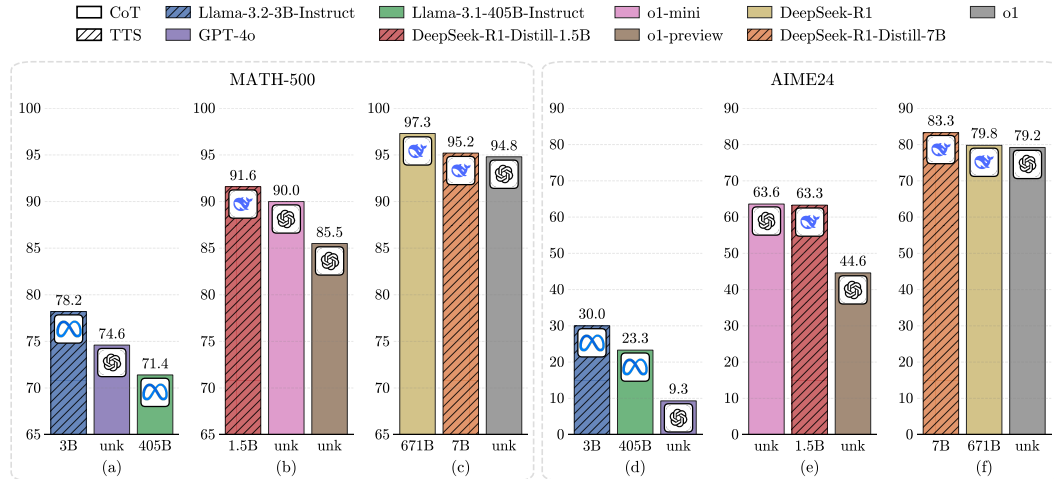


Figure 1: Comparison between the performance of smaller LLMs compute-optimal TTS and that of larger LLMs CoT on MATH-500 and AIME24. (a) & (d) Llama-3.2-3B-Instruct surpasses Llama-3.1-405B-Instruct and GPT-4o on MATH-500 and AIME24; (b) & (e) DeepSeek-R1-Distill-1.5B outperforms o1-preview on MATH-500 and AIME24, and surpasses o1-mini on MATH-500; (c) & (f) DeepSeek-R1-Distill-7B beats o1 on MATH-500 and AIME24, and exceeds DeepSeek-R1 on AIME24.

scale compute optimally, that is, allocating the optimal computation for each problem (Snell et al., 2024). Current TTS methods guide the generation process and select the final answer using Process Reward Models (PRMs), which effectively scale test-time compute (Wu et al., 2024; Snell et al., 2024; Beeching et al., 2024). These TTS methods involve several important factors, such as policy models¹, PRMs, and problem difficulty levels. However, there is limited systematic analysis of how policy models, PRMs, and problem difficulty influence these TTS strategies. This limitation prevents the community from fully understanding the effectiveness of this method and developing insights for compute-optimal TTS strategies.

To address these issues, this paper aims to investigate the influence of policy models, PRMs, and problem difficulty on TTS through comprehensive experimental analysis. Furthermore, we explore the concrete characteristics and performance boundaries of TTS methods. Specifically, we conduct extensive experiments on MATH-500 (Lightman et al., 2024) and the challenging AIME24 (AI-MO, 2024) tasks using a range of PRMs (spanning from 1.5B to 72B across different model series) across multiple policy models (ranging from 0.5B to 72B across two model families). Our results show that the compute-optimal TTS strategy heavily depends on the specific policy model, PRM, and problem difficulty level. Even smaller models (e.g., a 1B model) can outperform larger models (e.g., a 405B model) and even state-of-the-art reasoning models, such as o1 or DeepSeek-R1, in challenging reasoning tasks by applying compute-optimal TTS.

The contributions of this work can be summarized as follows:

¹Following Snell et al. (2024), we use “policy models” to refer to LLMs that generate solutions, and “verifiers” for PRMs.

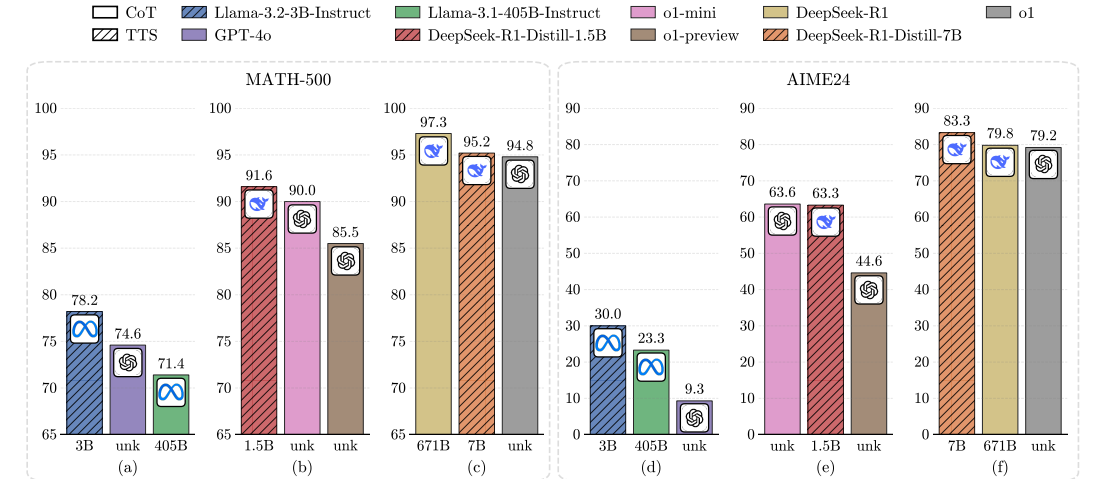


Figure 1: 在MATH-500和AIME24上，较小的LLM计算最优TTS与较大的LLM CoT性能的比较。(a) & (d) Llama-3.2-3B-Instruct 在MATH-500和AIME24上超过了Llama-3.1-405B-Instruct和GPT-4o; (b) & (e) DeepSeek-R1-Distill-1.5B 在MATH-500和AIME24上优于o1-preview，并且在MATH-500上超过了o1-mini; (c) & (f) DeepSeek-R1-Distill-7B 在MATH-500和AIME24上击败了o1，并且在AIME24上超过了DeepSeek-R1。

al., 2024; Beeching et al., 2024)。这些TTS方法涉及几个重要因素，如策略模型¹、PRMs和问题难度。然而，对于策略模型、PRMs和问题难度如何影响这些TTS策略，目前缺乏系统的分析。这一局限性阻碍了社区对这种方法的有效性进行全面理解，并开发出计算最优的TTS策略的见解。

为了解决这些问题，本文旨在通过全面的实验分析来研究策略模型、PRMs和问题难度对TTS的影响。此外，我们探索了TTS方法的具体特性和性能边界。具体而言，我们使用一系列PRMs（从1.5B到72B，涵盖不同的模型系列）和多个策略模型（从0.5B到72B，涵盖两个模型家族），在MATH-500 (Lightman et al., 2024) 和具有挑战性的AIME24 (AI-MO, 2024) 任务上进行了广泛的实验。我们的结果显示，计算最优的TTS策略高度依赖于特定的策略模型、PRM和问题难度水平。即使较小的模型（例如，1B 模型）在应用计算最优的TTS时，也可以在具有挑战性的推理任务中超越较大的模型（例如，405B 模型）甚至最先进的推理模型，如 o1 或 DeepSeek-R1。

本工作的贡献可以总结如下：

1. 我们对不同的TTS方法进行了全面评估，使用了各种最新的策略模型、多个PRM、多样的缩放方法和更具挑战性的任务。
2. 我们的分析强调了在TTS过程中考虑奖励影响的必要性，并引入了奖励感知的计算最优TTS。我们还展示了计算最优的扩展策略会随着不同的策略模型、PRMs和问题难度级别的变化而变化。
3. 实证结果表明，较小的语言模型通过TTS具有显著的超越较大模型的潜力。使用奖励感知的计算最优TTS策略，我们展示了3B LLM可以超越405B LLM，而7B LLM可以在MATH-500和AIME24任

¹根据 Snell et al. (2024)，我们使用“策略模型”来指代生成解决方案的LLMs，而“验证器”则指代PRMs。

1. We conduct a comprehensive evaluation of different TTS methods using various up-to-date policy models, multiple PRMs, diverse scaling methods, and more challenging tasks.
2. Our analysis highlights the necessity of considering the influence of rewards in the TTS process and introduces reward-aware compute-optimal TTS. We also demonstrate that the compute-optimal scaling strategy varies with different policy models, PRMs, and problem difficulty levels.
3. The empirical results demonstrate the significant potential of smaller language models to outperform larger models through TTS. Using the reward-aware Compute-optimal TTS strategy, we show that a **3B** LLM can outperform a **405B** LLM, and a **7B** LLM can surpass **o1** and **DeepSeek-R1** on MATH-500 and AIME24 tasks.

2. Setup & Preliminaries

2.1. Problem Formulation

We formulate the reasoning problem as a Markov Decision Process (MDP) (Sutton and Barto, 2018), defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where \mathcal{S} is the state space, \mathcal{A} is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the transition function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor. Given a prompt $x \sim \mathcal{X}$, the policy with parameters θ generates the initial action $a_1 \sim \pi_\theta(\cdot | s_1)$, where $s_1 = x$ is the initial state. The policy receives a reward $\mathcal{R}(s_1, a_1)$, and the state transitions to $s_2 = [s_1, a_1]$, where $[\cdot, \cdot]$ denotes the concatenation of two strings. This process continues until the episode terminates, either by reaching the maximum number of steps or by generating an `<EOS>` token. A trajectory of length H is represented as $\tau = \{a_1, a_2, \dots, a_H\}$. The process can be summarized as follows:

$$\begin{aligned}
 \text{Initial State:} \quad & s_1 = x \sim \mathcal{X} \\
 \text{Action:} \quad & a_t \sim \pi_\theta(\cdot | s_t) \\
 \text{State Transition:} \quad & s_{t+1} = \mathcal{P}(\cdot | s_t, a_t) = [s_t, a_t] \\
 \text{Reward:} \quad & r_t = \mathcal{R}(s_t, a_t)
 \end{aligned} \tag{1}$$

2.2. Test-Time Scaling Method

We consider three TTS methods: Best-of-N (BoN) (Brown et al., 2024), beam search (Snell et al., 2024), and Diverse Verifier Tree Search (DVTS) (Beeching et al., 2024). As pointed out by Snell et al. (2024), lookahead search is inefficient due to multi-step sampling, so we do not evaluate it or other methods involving lookahead operations, such as Monte Carlo Tree Search (MCTS). The TTS methods are shown in Figure 2.

Best-of-N. In the BoN approach, the policy model generates N responses, after which scoring and voting methods are applied to select the final answer.

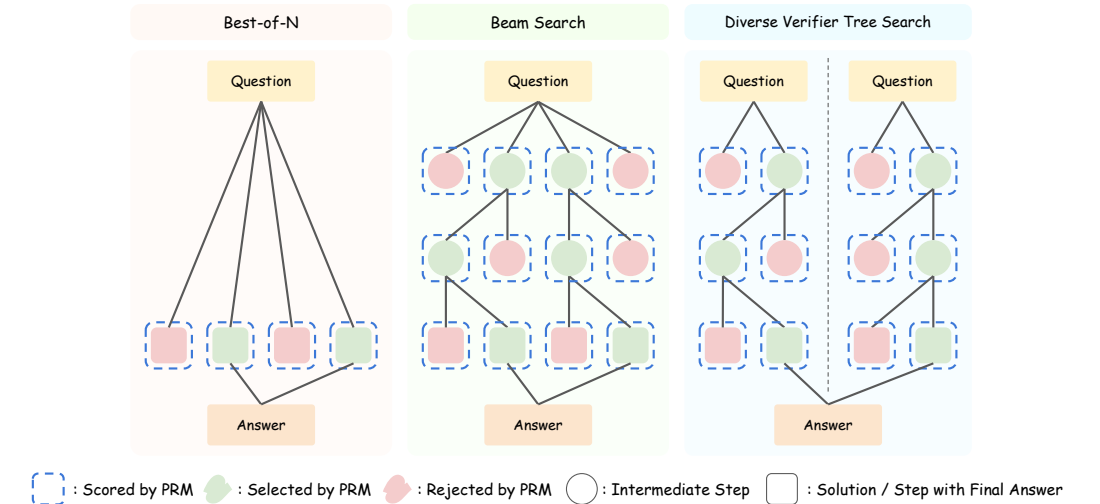


Figure 2: 不同外部TTS方法的比较。

务上超越o1和DeepSeek-R1。

2. Setup & Preliminaries

2.1. Problem Formulation

我们将推理问题表述为一个马尔可夫决策过程 (MDP) (Sutton and Barto, 2018)，由元组 $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ 定义，其中 \mathcal{S} 是状态空间， \mathcal{A} 是动作空间， $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ 是转移函数， $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ 是奖励函数， $\gamma \in [0, 1]$ 是折扣因子。给定一个提示 $x \sim \mathcal{X}$ ，具有参数 θ 的策略生成初始动作 $a_1 \sim \pi_\theta(\cdot | s_1)$ ，其中 $s_1 = x$ 是初始状态。策略接收到奖励 $\mathcal{R}(s_1, a_1)$ ，状态转换为 $s_2 = [s_1, a_1]$ ，其中 $[\cdot, \cdot]$ 表示两个字符串的连接。此过程持续进行，直到剧集终止，无论是达到最大步数还是生成一个 `<EOS>` 标记。长度为 H 的轨迹表示为 $\tau = \{a_1, a_2, \dots, a_H\}$ 。该过程可以概括如下：

$$\begin{aligned}
 \text{Initial State:} \quad & s_1 = x \sim \mathcal{X} \\
 \text{Action:} \quad & a_t \sim \pi_\theta(\cdot | s_t) \\
 \text{State Transition:} \quad & s_{t+1} = \mathcal{P}(\cdot | s_t, a_t) = [s_t, a_t] \\
 \text{Reward:} \quad & r_t = \mathcal{R}(s_t, a_t)
 \end{aligned} \tag{1}$$

2.2. Test-Time Scaling Method

我们考虑了三种TTS方法：Best-of-N (BoN) (Brown et al., 2024)、束搜索 (Snell et al., 2024)和Diverse Verifier Tree Search (DVTS) (Beeching et al., 2024)。正如 Snell et al. (2024)所指出的，由于多步采样的原因，前瞻搜索效率低下，因此我们不评估它或其他涉及前瞻操作的方法，如蒙特卡洛树搜索 (MCTS)。TTS方法如图 2所示。

最佳-N. 在BoN方法中，策略模型生成 N 个响应，之后应用评分和投票方法来选择最终答案。

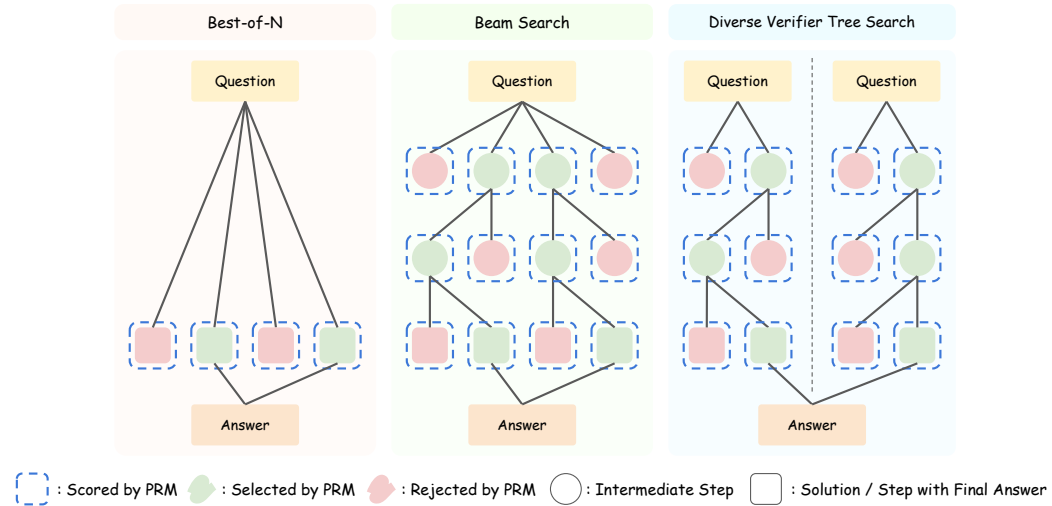


Figure 2: Comparison of different external TTS methods.

Beam Search. Given beam width N and beam size M , the policy model first generates N steps. The verifier selects the top $\frac{N}{M}$ steps for subsequent search. In the next step, the policy model samples M steps for each selected previous step. This process repeats until the maximum depth is reached or an $\langle \text{EOS} \rangle$ token is generated.

Diverse Verifier Tree Search. To increase diversity, DVTS extends beam search by dividing the search process into $\frac{N}{M}$ subtrees, each of which is explored independently using beam search. As shown in Beeching et al. (2024), DVTS outperforms beam search on easy and medium problems with a large computational budget N . A similar trend is observed in Chen et al. (2024), where increasing the number of parallel subtrees proves to be more effective than increasing the beam width under the same budget.

2.3. Compute-Optimal Test-Time Scaling

To maximize the performance of TTS, Snell et al. (2024) proposes a test-time compute-optimal scaling strategy, which selects hyperparameters corresponding to a given test-time strategy to maximize performance benefits on a specific prompt. Given a prompt x , let $\text{Target}(\theta, N, x)$ represent the output distribution over x produced by the policy model with parameters θ and a compute budget of N .

$$\theta_{x, y^*(x)}^*(N) = \arg \max_{\theta} (\mathbb{E}_{y \sim \text{Target}(\theta, N, x)} [\mathbb{1}_{y=y^*(x)}]), \quad (2)$$

where $y^*(x)$ denotes the ground-truth correct response for x , and $\theta_{x, y^*(x)}^*(N)$ represents the test-time compute-optimal scaling strategy for the problem x with compute budget N .

束搜索. 给定束宽 N 和束大小 M , 策略模型首先生成 N 个步骤。验证器选择前 $\frac{N}{M}$ 个步骤进行后续搜索。在下一步中, 策略模型为每个选定的前一步骤采样 M 个步骤。此过程重复进行, 直到达到最大深度或生成 $\langle \text{EOS} \rangle$ 标记。

多样化验证器树搜索. 为了增加多样性, DVTS通过将搜索过程分为 $\frac{N}{M}$ 个子树来扩展束搜索, 每个子树使用束搜索独立探索。如 Beeching et al. (2024)所示, DVTS在计算预算 N 较大时, 在简单和中等难度问题上优于束搜索。在 Chen et al. (2024)中也观察到类似趋势, 即在相同预算下, 增加并行子树的数量比增加束宽更有效。

2.3. Compute-Optimal Test-Time Scaling

为了最大化TTS的性能, Snell et al. (2024) 提出了一种测试时计算最优的缩放策略, 该策略选择与给定测试时策略相对应的超参数, 以最大化在特定提示下的性能收益。给定提示 x , 令 $\text{Target}(\theta, N, x)$ 表示参数为 θ 且计算预算为 N 的策略模型在 x 上产生的输出分布。

$$\theta_{x, y^*(x)}^*(N) = \arg \max_{\theta} (\mathbb{E}_{y \sim \text{Target}(\theta, N, x)} [\mathbb{1}_{y=y^*(x)}]), \quad (2)$$

其中 $y^*(x)$ 表示 x 的真实正确响应, 而 $\theta_{x, y^*(x)}^*(N)$ 表示在计算预算为 N 的情况下, 问题 x 的测试时计算最优缩放策略。

3. Rethinking Compute-Optimal Test-Time Scaling

3.1. Compute-Optimal Scaling Strategy Should be Reward-Aware

计算最优的TTS旨在为每个问题分配最优的计算资源 (Snell et al., 2024)。以往的TTS研究使用单个PRM作为验证器 (Snell et al., 2024; Wu et al., 2024; Beeching et al., 2024)。Snell et al. (2024)在一个策略模型的响应上训练了一个PRM, 并用它作为验证器与同一个策略模型一起进行TTS, 而 Wu et al. (2024); Beeching et al. (2024)使用在不同策略模型上训练的PRM进行TTS。从强化学习 (RL) 的角度来看, 前者获得的是在线 PRM, 后者获得的是离线 PRM。在线PRM为策略模型的响应生成更准确的奖励, 而离线PRM由于分布外 (OOD) 问题通常生成不准确的奖励 (Snell et al., 2024; Zheng et al., 2024)。

在实际应用计算最优的TTS时, 为每个策略模型训练一个PRM以防止OOD问题是计算成本高昂的。因此, 我们在一个更一般的设置中研究计算最优的TTS策略, 即PRM可能是在与用于TTS的策略模型不同的模型上训练的。对于基于搜索的方法, PRMs指导每个响应步骤的选择, 而对于基于采样的方法, PRMs在生成后评估响应。这表明 (1) 奖励影响所有方法的响应选择; (2) 对于基于搜索的方法, 奖励还影响搜索过程。

为了分析这些点, 我们使用Llama-3.1-8B-Instruct作为策略模型, RLHFlow-PRM-Mistral-8B和RLHFlow-PRM-Deepseek-8B作为PRMs, 进行了一项初步的案例研究。图 12中的结果显示, 奖励显著影响生

3. Rethinking Compute-Optimal Test-Time Scaling

3.1. Compute-Optimal Scaling Strategy Should be Reward-Aware

Compute-optimal TTS aims to allocate the optimal compute for each problem (Snell et al., 2024). Previous works on TTS use a single PRM as verifier (Snell et al., 2024; Wu et al., 2024; Beeching et al., 2024). Snell et al. (2024) trains a PRM on the responses of a policy model and uses it as the verifier to do TTS with the same policy model, while Wu et al. (2024); Beeching et al. (2024) use a PRM trained on a different policy model to do TTS. From the perspective of Reinforcement Learning (RL), we obtain an *on-policy* PRM in the former case and an *offline* PRM in the latter case. The on-policy PRM produces more accurate rewards for the responses of the policy model, while the offline PRM often generates inaccurate rewards due to out-of-distribution (OOD) issues (Snell et al., 2024; Zheng et al., 2024).

For practical applications of compute-optimal TTS, training a PRM for each policy model to prevent OOD issues is computationally expensive. Therefore, we investigate the compute-optimal TTS strategy in a more general setting, where the PRM might be trained on a different policy model than the one used for TTS. For search-based methods, PRMs guide the selection at each response step, while for sampling-based methods, PRMs evaluate the responses after generation. This indicates that (1) the reward influences response selection across all methods; (2) for search-based methods, the reward also influences the search process.

To analyze these points, we perform a preliminary case study using beam search with Llama-3.1-8B-Instruct as the policy model and RLHFlow-PRM-Mistral-8B and RLHFlow-PRM-Deepseek-8B as PRMs. The results in Figure 12 demonstrate that the reward significantly affects the generation process and outcomes. RLHFlow-PRM-Mistral-8B assigns high rewards to short responses, leading to incorrect answers, while searching with RLHFlow-Deepseek-PRM-8B produces correct answers but uses more tokens. In Section 4, we also empirically show that rewards have great influence on TTS performance and output tokens.

Based on these findings, we propose that *rewards* should be integrated into the compute-optimal TTS strategy. Let us denote the reward function as \mathcal{R} . Our reward-aware compute-optimal TTS strategy is formulated as:

$$\theta_{x,y^*(x),\mathcal{R}}^*(N) = \arg \max_{\theta} (\mathbb{E}_{y \sim \text{Target}(\theta, N, x, \mathcal{R})} [\mathbb{1}_{y=y^*(x)}]), \quad (3)$$

where $\text{Target}(\theta, N, x, \mathcal{R})$ represents the output distribution of the policy model θ , adjusted by the reward function \mathcal{R} , under a compute budget N and prompt x . For sampling-based scaling methods, $\text{Target}(\theta, N, x, \mathcal{R}) = \text{Target}(\theta, N, x)$. This reward-aware strategy ensures that compute-optimal scaling adapts to the policy model, prompt, and reward function, leading to a more general framework for practical TTS.

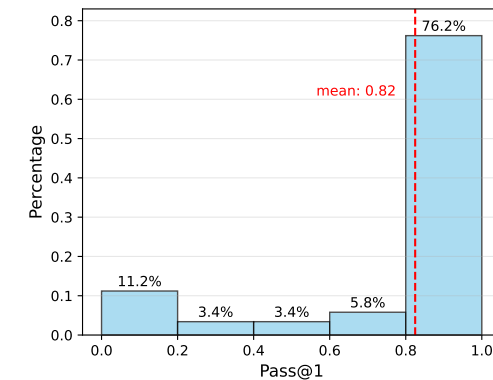


Figure 3: Qwen2.5-72B-Instruct 在 MATH-500 上的 Pass@1 准确率分布，分为五个区间。

成过程和结果。RLHFlow-PRM-Mistral-8B对短响应给予高奖励，导致错误答案，而使用RLHFlow-Deepseek-PRM-8B进行搜索则产生正确答案但使用了更多令牌。在第 4 节中，我们还通过实验证明了奖励对TTS性能和输出令牌有重大影响。

基于这些发现，我们提出应将奖励整合到计算最优的TTS策略中。让我们将奖励函数表示为 \mathcal{R} 。我们的奖励感知计算最优TTS策略表述如下：

$$\theta_{x,y^*(x),\mathcal{R}}^*(N) = \arg \max_{\theta} (\mathbb{E}_{y \sim \text{Target}(\theta, N, x, \mathcal{R})} [\mathbb{1}_{y=y^*(x)}]), \quad (3)$$

其中 $\text{Target}(\theta, N, x, \mathcal{R})$ 表示在计算预算 N 和提示 x 下，由奖励函数 \mathcal{R} 调整的策略模型 θ 的输出分布。对于基于采样的扩展方法， $\text{Target}(\theta, N, x, \mathcal{R}) = \text{Target}(\theta, N, x)$ 。这种奖励感知策略确保计算最优扩展适应于策略模型、提示和奖励函数，从而为实际的 TTS 提供了一个更通用的框架。

3.2. Absolute Problem Difficulty Criterion is More Effective Than Quantiles

为了考虑问题难度对 TTS 的影响，Snell et al. (2024) 根据 Pass@1 准确率的分位数将问题分为五个难度级别。然而，我们发现使用 MATH (Hendrycks et al., 2021) 的难度级别或基于 Pass@1 准确率分位数的 oracle 标签 (Snell et al., 2024) 并不有效，因为不同的策略模型具有不同的推理能力。如图 3 所示，Qwen2.5-72B-Instruct 在 76.2% 的 MATH-500 问题上实现了超过 80% 的 Pass@1 准确率。因此，我们使用绝对阈值而不是分位数来衡量问题难度。具体来说，我们根据 Pass@1 准确率定义了三个难度级别：简单（50% ~ 100%）、中等（10% ~ 50%）和困难（0% ~ 10%）。

4. How to Scale Test-Time Compute Optimally?

在本节中，我们旨在回答以下问题：

- **Q1:** TTS 如何通过不同的策略模型和PRMs得到改进？
- **Q2:** TTS 如何在不同难度的问题上得到改进？
- **Q3:** PRMs是否对特定的回答长度有偏见或对投票方法敏感？

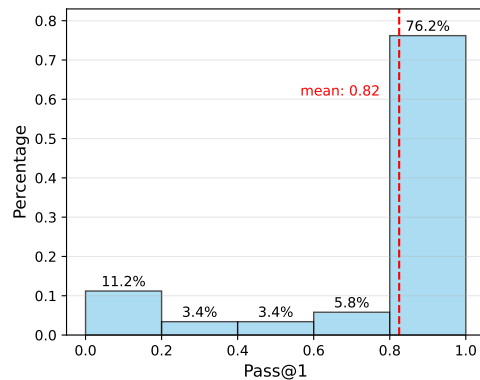


Figure 3: Distribution of Pass@1 accuracy of Qwen2.5-72B-Instruct on MATH-500, divided into five bins.

3.2. Absolute Problem Difficulty Criterion is More Effective Than Quantiles

To consider the influence of problem difficulty on TTS, Snell et al. (2024) group problems into five difficulty levels based on Pass@1 accuracy quantiles. However, we find that using difficulty levels from MATH (Hendrycks et al., 2021) or oracle labels based on Pass@1 accuracy quantiles (Snell et al., 2024) is not effective since different policy models have different reasoning capabilities. As shown in Figure 3, Qwen2.5-72B-Instruct achieves Pass@1 accuracy above 80% on 76.2% of MATH-500 problems. Therefore, we use absolute thresholds instead of quantiles to measure problem difficulty. Specifically, we define three difficulty levels based on Pass@1 accuracy: easy (50% ~ 100%), medium (10% ~ 50%), and hard (0% ~ 10%).

4. How to Scale Test-Time Compute Optimally?

In this section, we aim to answer the following questions:

- **Q1:** How does TTS improve with different policy models and PRMs?
- **Q2:** How does TTS improve for problems with different difficulty levels?
- **Q3:** Does PRMs have bias towards specific response lengths or sensitivity to voting methods?

4.1. Setup

Datasets. We conduct experiments on competition-level mathematical datasets, including MATH-500 (Lightman et al., 2024) and AIME24 (AI-MO, 2024). MATH-500 contains 500 representative problems from the test set of MATH (Hendrycks et al., 2021), and this subset is used following Snell et al. (2024); Beeching et al. (2024). As recent LLMs show significant progress in mathematical reasoning (OpenAI, 2024; DeepSeek-AI et al., 2025), we include the more challenging AIME24 for experiments.

4.1. Setup

数据集. 我们在竞赛级别的数学数据集上进行实验，包括 MATH-500 (Lightman et al., 2024) 和 AIME24 (AI-MO, 2024)。MATH-500 包含从 MATH (Hendrycks et al., 2021) 测试集中选出的 500 个代表性问题，这个子集的使用遵循 Snell et al. (2024); Beeching et al. (2024)。由于最近的大型语言模型在数学推理方面取得了显著进展 (OpenAI, 2024; DeepSeek-AI et al., 2025)，我们加入了更具挑战性的 AIME24 用于实验。

策略模型. 对于测试时的方法，我们使用来自 Llama 3 (Dubey et al., 2024) 和 Qwen2.5 (Yang et al., 2024b) 系列的不同大小的策略模型。我们对所有策略模型使用 *Instruct* 版本。

过程奖励模型. 我们考虑以下开源 PRMs 进行评估：

- **Math-Shepherd (Wang et al., 2024b):** Math-Shepherd-PRM-7B 是在 Mistral-7B (Jiang et al., 2023) 上使用从 Mistral-7B 在 MetaMath (Yu et al., 2024) 上微调后生成的 PRM 数据训练的。
- **RLHFlow 系列 (Xiong et al., 2024):** RLHFlow 包括 RLHFlow-PRM-Mistral-8B 和 RLHFlow-PRM-Deepseek-8B，它们分别基于 Mistral-7B 在 MetaMath (Yu et al., 2024) 和 deepseek-math-7b-instruct (Shao et al., 2024) 上微调的数据进行训练。这两种 PRMs 的基础模型都是 Llama-3.1-8B-Instruct (Dubey et al., 2024)。
- **Skywork 系列 (Skywork o1 Team, 2024):** Skywork 系列包括 Skywork-PRM-1.5B 和 Skywork-PRM-7B，分别基于 Qwen2.5-Math-1.5B-Instruct 和 Qwen2.5-Math-7B-Instruct (Yang et al., 2024c) 训练。训练数据来自在数学数据集上微调的 Llama-2 (Touvron et al., 2023) 和 Qwen2-Math (Yang et al., 2024a) 系列模型。
- **Qwen2.5-Math 系列 (Zhang et al., 2025):** 我们评估了 Qwen2.5-Math-PRM-7B 和 Qwen2.5-Math-PRM-72B，它们分别基于 Qwen2.5-Math-7B-Instruct 和 Qwen2.5-Math-72B-Instruct (Yang et al., 2024c) 训练。训练数据是使用 Qwen2-Math (Yang et al., 2024a) 和 Qwen2.5-Math 系列模型 (Yang et al., 2024c) 生成的。在所有列出的 PRMs 中，Qwen2.5-Math-PRM-72B 是最强的开源 PRM 用于数学任务，而 Qwen2.5-Math-PRM-7B 是参数量为 7B/8B 的 PRMs 中最强大的，如 Zhang et al. (2025) 所示。

评分和投票方法. 根据 Wang et al. (2024a)，我们考虑了三种评分方法：PRM-Min，PRM-Last，和 PRM-Avg，以及三种投票方法：多数投票，PRM-Max，和 PRM-Vote。为了获得最终答案，我们首先使用评分方法来评估答案。对于长度为 H 的轨迹，使用不同的评分方法计算每个轨迹的分数如下：(1) PRM-Min 通过所有步骤中的最小奖励来评分，即 $\text{score} = \min_{t=0}^H \mathcal{R}_t$ 。(2) PRM-Last 通过最后一步的奖励来评分，即 $\text{score} = \mathcal{R}_H$ 。(3) PRM-Avg 通过所有步骤的平均奖励来评分，即 $\text{score} = \frac{1}{H} \sum_{t=0}^H \mathcal{R}_t$ 。投票方法随后汇总分数以确定最终答案。多数投票 选择获得多数票的答案 (Wang et al., 2023)，而 PRM-Max 选择得分最高的答案，PRM-Vote 首先累积所有相同答案的分数，然后选择得分最高的答案。

Policy Models. For test-time methods, we use policy models from Llama 3 (Dubey et al., 2024) and Qwen2.5 (Yang et al., 2024b) families with different sizes. We use the *Instruct* version for all policy models.

Process Reward Models. We consider the following open-source PRMs for evaluation:

- **Math-Shepherd** (Wang et al., 2024b): **Math-Shepherd-PRM-7B** is trained on Mistral-7B (Jiang et al., 2023) using PRM data generated from Mistral-7B fine-tuned on MetaMath (Yu et al., 2024).
- **RLHFlow Series** (Xiong et al., 2024): RLHFlow includes **RLHFlow-PRM-Mistral-8B** and **RLHFlow-PRM-Deepseek-8B**, which are trained on data from Mistral-7B fine-tuned on MetaMath (Yu et al., 2024) and deepseek-math-7b-instruct (Shao et al., 2024), respectively. The base model for both PRMs is Llama-3.1-8B-Instruct (Dubey et al., 2024).
- **Skywork Series** (Skywork o1 Team, 2024): The Skywork series comprises **Skywork-PRM-1.5B** and **Skywork-PRM-7B**, trained on Qwen2.5-Math-1.5B-Instruct and Qwen2.5-Math-7B-Instruct (Yang et al., 2024c), respectively. The training data is generated from Llama-2 (Touvron et al., 2023) fine-tuned on a mathematical dataset and Qwen2-Math (Yang et al., 2024a) series models.
- **Qwen2.5-Math Series** (Zhang et al., 2025): We evaluate **Qwen2.5-Math-PRM-7B** and **Qwen2.5-Math-PRM-72B**, trained on Qwen2.5-Math-7B-Instruct and Qwen2.5-Math-72B-Instruct (Yang et al., 2024c), respectively. The data for training is generated using Qwen2-Math (Yang et al., 2024a) and Qwen2.5-Math series models (Yang et al., 2024c). Among all the PRMs listed, Qwen2.5-Math-PRM-72B is the strongest open-source PRM for mathematical tasks, while Qwen2.5-Math-PRM-7B is the most capable PRM among those with 7B/8B parameters, as demonstrated in Zhang et al. (2025).

Scoring and Voting Methods. Following Wang et al. (2024a), we consider three scoring methods: *PRM-Min*, *PRM-Last*, and *PRM-Avg*, and three voting methods: *Majority Vote*, *PRM-Max*, and *PRM-Vote*. To obtain the final answer, we first use the scoring methods to evaluate the answers. For a trajectory of length H , the scores for each trajectory with different scoring methods are calculated as follows: (1) *PRM-Min* scores each trajectory by the minimum reward among all steps, i.e., $\text{score} = \min_{\mathcal{R}} \{\mathcal{R}_t\}_{t=0}^H$. (2) *PRM-Last* scores each trajectory by the reward of the last step, i.e., $\text{score} = \mathcal{R}_H$. (3) *PRM-Avg* scores each trajectory by the average reward among all steps, i.e., $\text{score} = \frac{1}{H} \sum_{t=0}^H \mathcal{R}_t$. The voting methods then aggregate the scores to determine the final answer. *Majority Vote* selects the answer with the majority of votes (Wang et al., 2023), while *PRM-Max* selects the answer with the highest score, and *PRM-Vote* first accumulates the scores of all identical answers and then selects the answer with the highest score.

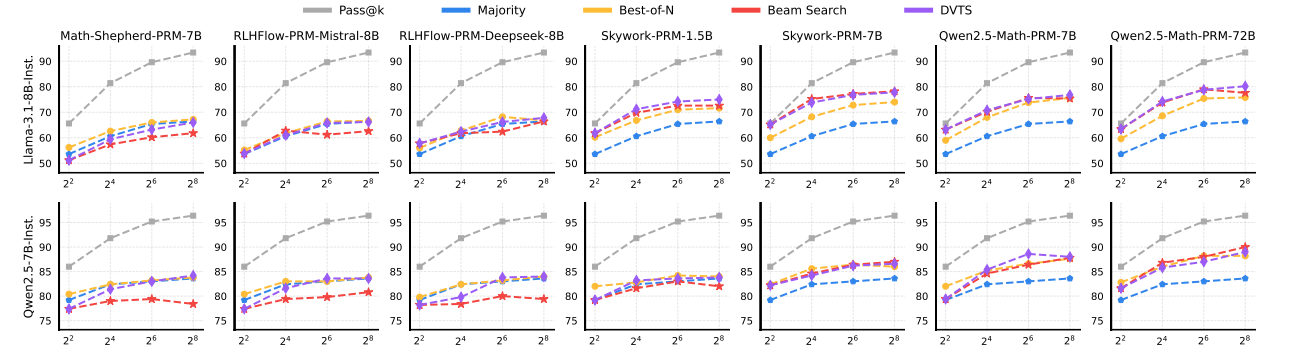


Figure 4: Llama-3.1-8B-Instruct 和 Qwen2.5-7B-Instruct 在 MATH-500 上使用不同的 PRMs 和 TTS 策略的性能。

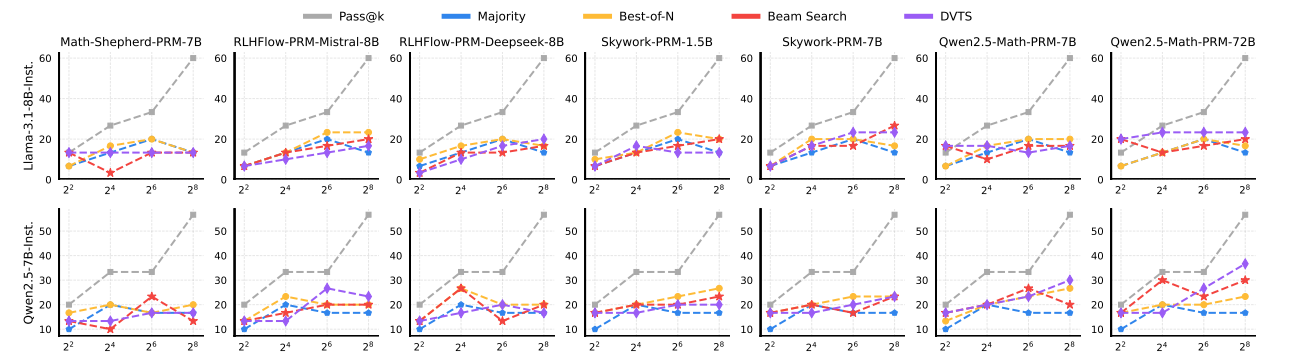


Figure 5: Llama-3.1-8B-Instruct 和 Qwen2.5-7B-Instruct 在 AIME24 上使用不同的 PRMs 和 TTS 策略的性能。

我们使用 OpenR², 这是一个开源的 LLM 推理框架作为我们的代码库。对于计算预算, 我们在大多数实验中使用 $\{4, 16, 64, 256\}$ 。步骤的划分遵循先前工作 (Xiong et al., 2024; Zhang et al., 2025) 中的格式 $\backslash n \backslash n$ 。对于束搜索和 DVTs, 束宽度设置为 4。CoT 的温度为 0.0, 而其他方法的温度为 0.7。对于 CoT 和 BoN, 我们限制新生成的令牌的最大数量为 8192。对于基于搜索的方法, 每步的令牌限制为 2048, 总响应的令牌限制为 8192。

4.2. How does TTS improve with different policy models and PRMs? (Q1)

PRMs 在不同策略模型和任务之间难以泛化。如图 4 所示, 对于 Llama-3.1-8B-Instruct, 使用 Skywork 和 Qwen2.5-Math PRMs 的基于搜索的方法在更大的计算预算下性能显著提高, 而使用 Math-Shepherd 和 RLHFlow PRMs 的搜索结果仍然相对较差, 甚至不如多数投票。对于 Qwen2.5-7B-Instruct, 使用 Skywork-PRM-7B 和 Qwen2.5-Math PRMs 的搜索性能随着更多预算的增加而表现良好, 而其他 PRMs 的性能仍然较差。在图 5 中, 尽管两种策略模型的 Pass@k 准确率随着更大的计算预算显著提高, 但 TTS 的性能提升仍然适中。这些结果表明, PRMs 在不同策略模型和任务之间的泛化特别具有挑战性, 尤其是对于更复杂的任务。

²<https://github.com/openreasoner/openr>

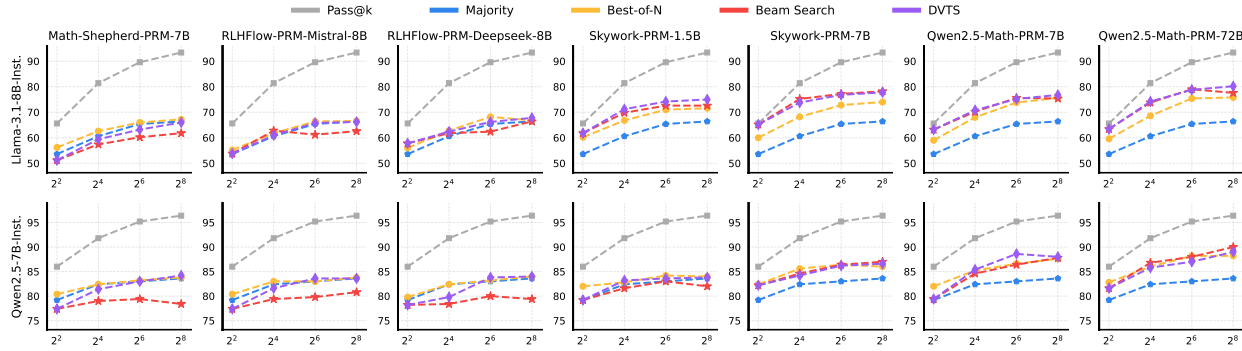


Figure 4: Performance of Llama-3.1-8B-Instruct and Qwen2.5-7B-Instruct on MATH-500 with different PRMs and TTS strategies.

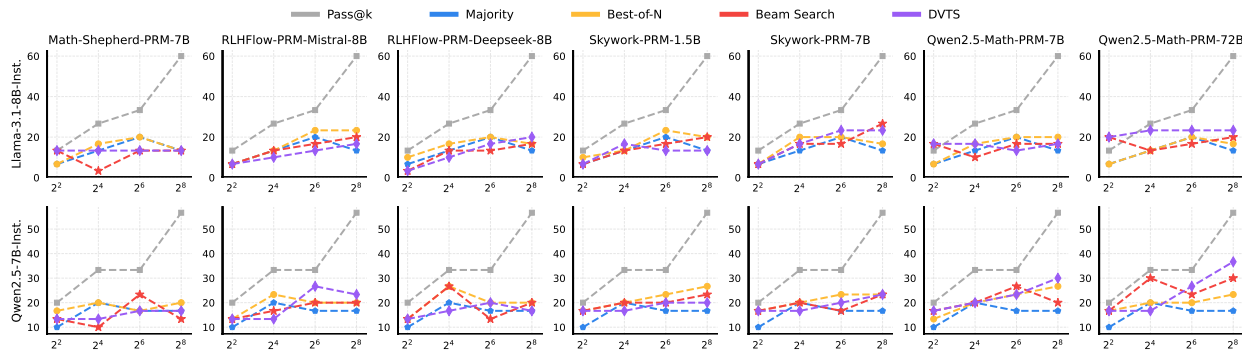


Figure 5: Performance of Llama-3.1-8B-Instruct and Qwen2.5-7B-Instruct on AIME24 with different PRMs and TTS strategies.

We use OpenR², which is an open-source LLM reasoning framework as our codebase. For compute budgets, we use $\{4, 16, 64, 256\}$ in most experiments. The division of steps follows the format $\backslash n \backslash n$ as in prior works (Xiong et al., 2024; Zhang et al., 2025). For beam search and DVTS, the beam width is set to 4. The temperature of CoT is 0.0, while it is 0.7 for other methods. For CoT and BoN, we restrict the maximum number of new tokens to 8192. For search-based methods, the token limit is 2048 for each step and 8192 for the total response.

4.2. How does TTS improve with different policy models and PRMs? (Q1)

PRMs are hard to generalize across policy models and tasks. As shown in Figure 4, for Llama-3.1-8B-Instruct, the performance of search-based methods with Skywork and Qwen2.5-Math PRMs improves significantly with larger compute budgets, while the results of searching with Math-Shepherd and RLHFlow PRMs remain relatively poor, even worse than majority voting. For Qwen2.5-7B-Instruct, the performance of searching with Skywork-PRM-7B and Qwen2.5-Math PRMs scales well with more budgets, while the performance of other PRMs remains poor. In Figure 5, although the Pass@k accuracy

²<https://github.com/openreasoner/openr>

最优的 TTS 方法取决于所使用的 PRM。如图 4 所示, 当使用 Math-Shepherd 和 RLHFlow PRMs 时, BoN 通常优于其他策略, 而使用 Skywork 和 Qwen2.5-Math PRMs 时, 基于搜索的方法表现更好。这种差异的发生是因为使用 PRM 对 OOD 策略响应进行处理会导致次优答案, 因为 PRMs 在不同策略模型之间的泛化能力有限。此外, 如果我们使用 OOD PRMs 选择 每一步, 可能会得到陷入局部最优的答案, 从而降低性能。这可能也与 PRM 的基础模型有关, 因为使用 PRM800K (Lightman et al., 2024) 训练的 Qwen2.5-Math-7B-Instruct PRM 泛化能力优于以 Mistral 和 Llama 为基础模型的 PRMs (Zhang et al., 2025)。进一步的分析见第 4.4 节和附录 C。这些结果表明, 最优 TTS 策略的选择取决于具体的 PRMs, 强调了在计算最优 TTS 中考虑奖励信息的重要性。我们还探讨了 TTS 性能与不同 PRMs 的过程监督能力之间的关系。如图 6 所示, TTS 性能与 PRMs 的过程监督能力呈正相关, 拟合函数为 $Y = 7.66 \log(X) + 44.31$, 其中 Y 表示 TTS 性能, X 表示 PRM 的过程监督能力 (Zhang et al., 2025)。

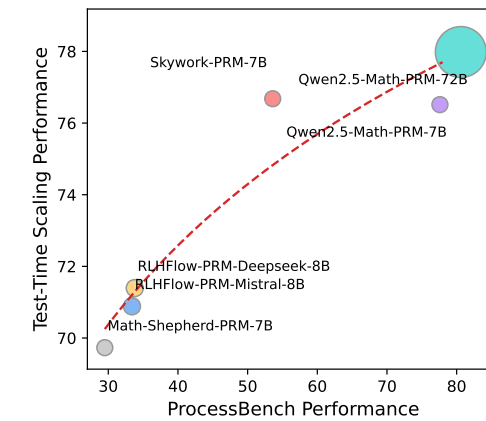


Figure 6: 不同PRMs在MATH上的TTS性能与过程监督能力之间的关系, 其中每个圆的大小代表PRM的参数数量, 曲线代表拟合函数。

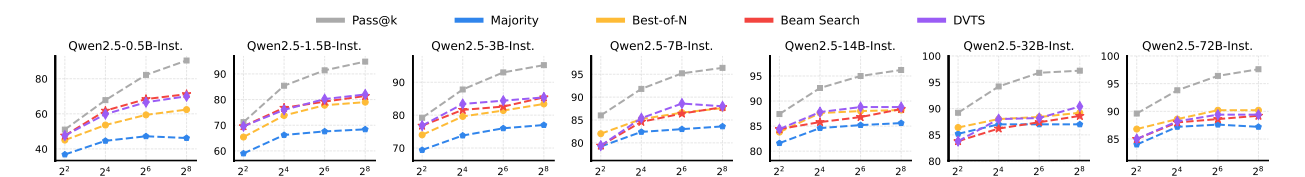


Figure 7: 具有0.5B到72B参数的策略模型在MATH-500上的TTS性能, 采用不同的缩放方法。

最优的TTS方法随策略模型而变化。为了研究策略模型参数与最优TTS方法之间的关系, 我们使用Qwen2.5家族LLMs (Yang et al., 2024b)进行了实验, 包括参数量为0.5B、1.5B、3B、7B、14B、32B和72B的模型。图 7 中的结果显示, 最优的TTS方法取决于具体的策略模型。对于小型策略模型, 基于搜索的方法优于BoN, 而对于大型策略模型, BoN比基于搜索的方法更有效。这种差异出现的原因是, 较大的模型具有更强的推理能力, 不需要验证器来进行逐步选择。相比之下, 较小的模型依赖于验证器来选择每一步, 以确保每个中间步骤的正确性。

of both policy models improves a lot with larger compute budgets, the performance improvement of TTS remains moderate. These results demonstrate that the generalization of PRMs is particularly challenging across different policy models and tasks, especially for more complex tasks.

The optimal TTS method depends on the PRM used. As shown in Figure 4, BoN outperforms other strategies most of the time when using Math-Shepherd and RLHFlow PRMs, while search-based methods perform better with Skywork and Qwen2.5-Math PRMs. This difference occurs because using a PRM for OOD policy responses leads to sub-optimal answers, as PRMs show limited generalization across policy models. Moreover, if we select *each step* with OOD PRMs, it is likely to obtain answers trapped in local optima and worsen the performance. This may also be related to the base model of the PRM, since the PRM trained with PRM800K (Lightman et al., 2024) on Qwen2.5-Math-7B-Instruct generalizes better than PRMs with Mistral and Llama as base models (Zhang et al., 2025). Further analysis is provided in Section 4.4 and Appendix C. These results suggest that the choice of the optimal TTS strategy depends on the specific PRMs used, emphasizing the importance of considering reward information in compute-optimal TTS. We also explore the relationship between TTS performance and the process supervision abilities of different PRMs. As shown in Figure 6, TTS performance is positively correlated with the process supervision abilities of PRMs, and the fitted function is $Y = 7.66 \log(X) + 44.31$, where Y represents TTS performance and X represents the process supervision abilities of the PRM (Zhang et al., 2025).

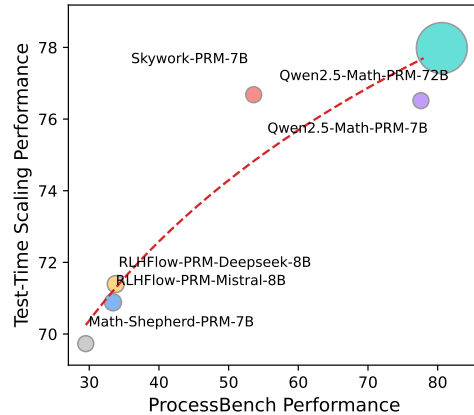


Figure 6: The relationship between TTS performance and process supervision abilities of different PRMs on MATH, where the size of each circle represents the number of parameters of the PRM and the curve represents the fitted function.

The optimal TTS method varies with policy models. To study the relationship between the parameters of the policy models and the optimal TTS methods, we conduct experiments with Qwen2.5 family LLMs (Yang et al., 2024b), including models with 0.5B, 1.5B, 3B, 7B, 14B, 32B, and 72B parameters. The results in Figure 7 show that the optimal TTS methods depend on the specific policy models. For small policy models, search-based methods outperform BoN, while for large policy models,

4.3. How does TTS improve for problems with different difficulty levels? (Q2)

根据 Snell et al. (2024), 我们对不同难度级别的任务进行了全面评估。然而, 正如在第 3.2 节中所解释的, 我们观察到使用 MATH (Hendrycks et al., 2021) 中定义的难度级别或基于 Pass@1 准确率的分位数的 oracle 标签是不合适的, 因为不同的策略模型表现出不同的推理能力。为了解决这个问题, 我们根据 Pass@1 准确率的绝对值将难度级别分为三组: 简单 (50% ~ 100%)、中等 (10% ~ 50%) 和困难 (0% ~ 10%)。

最优的 TTS 方法随不同的难度级别而变化。图 8 和图 9 的结果显示, 对于小型策略模型 (即参数少于 7B 的模型), BoN 在简单问题上表现更好, 而束搜索在更难的问题上表现更好。对于参数在 7B 到 32B 之间的策略模型, DVTS 在简单和中等问题上表现良好, 而束搜索在困难问题上更优。对于参数为 72B 的策略模型, BoN 是所有难度级别上的最佳方法。

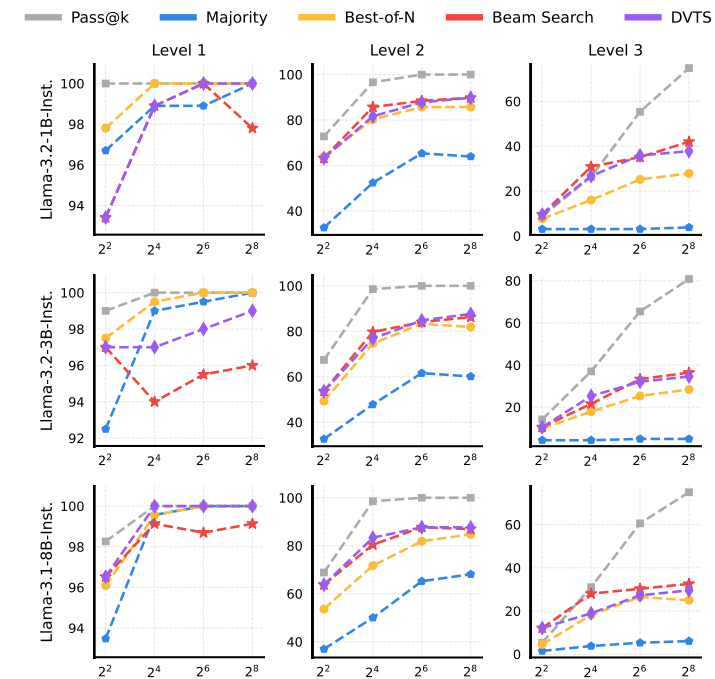


Figure 8: 三个 Llama 策略模型在 MATH-500 上的 TTS 性能, 包含三个难度级别。

4.4. Does PRMs have bias towards specific response lengths or sensitivity to voting methods? (Q3)

Table 1: RLHFlow PRMs 训练数据的统计信息。

	Mistral-PRM-Data	Deepseek-PRM-Data
Average Token per Response	236.9	333.1
Average Token per Step	46.6	58.4

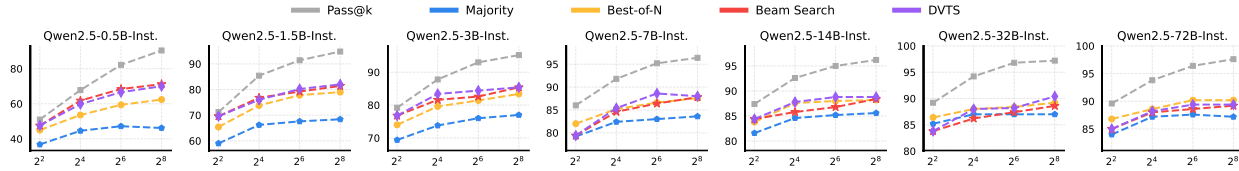


Figure 7: TTS performance of policy models with parameters from 0.5B to 72B on MATH-500 with different scaling methods.

BoN is more effective than search-based methods. This difference occurs because larger models have stronger reasoning capabilities and do not need a verifier to perform step-by-step selection. In contrast, smaller models rely on a verifier to select each step, ensuring the correctness of each intermediate step.

4.3. How does TTS improve for problems with different difficulty levels? (Q2)

Following Snell et al. (2024), we conduct a comprehensive evaluation of tasks with varying difficulty levels. However, as explained in Section 3.2, we observe that using the difficulty levels defined in MATH (Hendrycks et al., 2021) or the oracle labels based on the quantile of Pass@1 accuracy (Snell et al., 2024) is not appropriate because different policy models exhibit different reasoning abilities. To address this, we categorize the difficulty levels into three groups based on the absolute value of Pass@1 accuracy: easy (50% ~ 100%), medium (10% ~ 50%), and hard (0% ~ 10%).

The optimal TTS methods vary with different difficulty levels. The results in Figure 8 and Figure 9 show that for small policy models (i.e., with fewer than 7B parameters), BoN is better for easy problems, while beam search works better for harder problems. For policy models with parameters between 7B and 32B, DVTS performs well for easy and medium problems, and beam search is preferable for hard problems. For policy models with 72B parameters, BoN is the best method for all difficulty levels.

4.4. Does PRMs have bias towards specific response lengths or sensitivity to voting methods? (Q3)

Table 1: Statistics of training data of RLHFlow PRMs.

	Mistral-PRM-Data	Deepseek-PRM-Data
Average Token per Response	236.9	333.1
Average Token per Step	46.6	58.4

PRMs are biased towards the length of steps. Although we perform TTS under the same budget in pervious experiments, we find that the number of inference tokens with different PRMs varies significantly. For example, given the same budget and the same policy model, the number of inference

PRMs对步骤长度有偏见。 尽管我们在之前的实验中以相同的预算执行TTS，但我们发现不同PRMs的推理标记数量差异显著。例如，在相同的预算和相同的策略模型下，RLHFlow-PRM-Deepseek-8B的推理标记数量始终比RLHFlow-PRM-Mistral-8B多，几乎是后者的两倍。RLHFlow系列PRMs的训练数据是从不同的LLMs中采样的，这可能导致输出长度的偏差。为了验证这一点，我们分析了RLHFlow-PRM-Mistral-8B³和RLHFlow-PRM-Deepseek-8B⁴的训练数据的几个属性。如表 1所示，DeepSeek-PRM-Data的每响应平均标记数和每步平均标记数都大于Mistral-PRM-Data，表明RLHFlow-PRM-Deepseek-8B的训练数据比RLHFlow-PRM-Mistral-8B的训练数据更长。这可能导致输出长度的偏差。我们还发现，Qwen2.5-Math-7B的推理标记数量比Skywork-PRM-7B多，但性能非常接近，这表明使用Skywork-PRM-7B进行搜索比使用Qwen2.5-Math-7B更高效。

Table 2: 不同投票方法在MATH-500上的TTS性能。

	Skywork-PRM-7B	Qwen2.5-Math-PRM-7B
Majority Vote	86.8	87.6
PRM-Min-Max	83.0	87.4
PRM-Min-Vote	86.6	87.6
PRM-Last-Max	84.4	87.6
PRM-Last-Vote	87.0	87.6
PRM-Avg-Max	85.8	87.8
PRM-Avg-Vote	86.8	87.6

PRMs对投票方法敏感。 从表 2的结果可以看出，Skywork-PRM-7B 使用 *PRM-Vote* 比使用 *PRM-Max* 表现更好，而 Qwen2.5-Math-PRM-7B 对投票方法的敏感度不高。主要原因在于 Qwen2.5-Math PRMs的训练数据经过了 LLM-as-a-judge (Zheng et al., 2023) 处理，去除了训练数据中标记为正向步骤的错误中间步骤，使得输出的高奖励值更有可能是正确的。这表明 PRMs的训练数据对于提高搜索过程中发现错误的能力非常重要。

5. Results for Compute-Optimal Test-Time Scaling

在第 4节中探讨了计算最优的TTS策略后，我们进一步开展了实验以探索以下问题：

- **Q4:** 较小的策略模型是否可以通过计算最优的TTS策略超越较大的模型？
- **Q5:** 计算最优的TTS相比CoT和多数投票有何改进？
- **Q6:** TTS 方法是否比基于长链思考的方法更有效？

³<https://huggingface.co/datasets/RLHFlow/Mistral-PRM-Data>

⁴<https://huggingface.co/datasets/RLHFlow/Deepseek-PRM-Data>

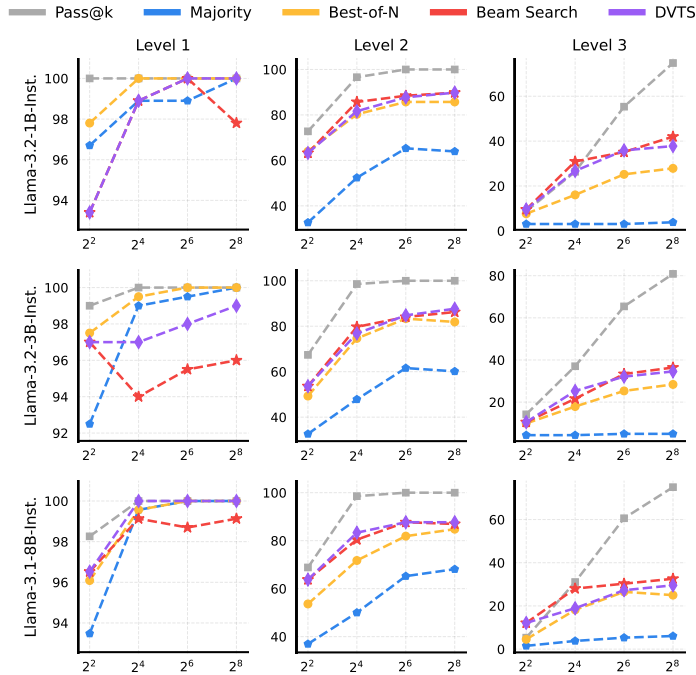


Figure 8: TTS performance of three Llama policy models on MATH-500 with three difficulty levels.

tokens of scaling with RLHFlow-PRM-Deepseek-8B is consistently larger than that of RLHFlow-PRM-Mistral-8B, nearly $2\times$. The training data of RLHFlow series PRMs are sampled from different LLMs, which may lead to the bias towards the length of the output. To verify this point, we analyze several properties of the training data of RLHFlow-PRM-Mistral-8B³ and RLHFlow-PRM-Deepseek-8B⁴. As shown in Table 1, both the average token per response and the average token per step of DeepSeek-PRM-Data are larger than those of Mistral-PRM-Data, indicating that the training data of RLHFlow-PRM-Deepseek-8B is longer than that of RLHFlow-PRM-Mistral-8B. This may lead to the bias towards the length of the output. We also find that the number of inference tokens of scaling with Qwen2.5-Math-7B is larger than that of Skywork-PRM-7B, but the performance is very near, which indicates that searching with Skywork-PRM-7B is more efficient than searching with Qwen2.5-Math-7B.

PRMs are sensitive to voting methods. From the results in Table 2, it is shown that Skywork-PRM-7B works better with *PRM-Vote* than with *PRM-Max*, while Qwen2.5-Math-PRM-7B is not very sensitive to voting methods. The main reason is that the training data of Qwen2.5-Math PRMs are processed with LLM-as-a-judge (Zheng et al., 2023), which removes the wrong intermediate steps labeled as positive steps in the training data and makes the outputted large reward values more likely to be correct. This shows that the training data of PRMs is important for improving the ability to find

³<https://huggingface.co/datasets/RLHFlow/Mistral-PRM-Data>

⁴<https://huggingface.co/datasets/RLHFlow/Deepseek-PRM-Data>

Table 3: 小规模策略模型（计算最优TTS）与前沿推理LLM（CoT）在MATH-500和AIME24上的比较。

Policy Model	MATH-500	AIME24	Avg.
<i>Proprietary LLMs (CoT)</i>			
GPT-4o	74.6	9.3	42.0
o1-preview	85.5	44.6	65.1
o1-mini	90.0	63.6	76.8
o1	94.8	79.2	87.0
<i>Open-Source LLMs (CoT)</i>			
Llama-3.1-70B-Inst.	65.2	16.7	41.0
Llama-3.1-405B-Inst.	71.4	23.3	47.4
QwQ-32B-Preview	90.6	50.0	70.3
DeepSeek-R1	97.3	79.8	88.6
<i>Open-Source LLMs (TTS)</i>			
Llama-3.2-1B-Inst.	66.2	16.7	41.5
Llama-3.2-1B-Inst. ($N = 512$)	72.2	10.0	41.1
Llama-3.2-3B-Inst.	75.6	30.0	52.8
Qwen2.5-0.5B-Inst.	76.4	10.0	43.2
Qwen2.5-1.5B-Inst.	81.8	20.0	50.9
DeepSeek-R1-Distill-Qwen-1.5B	91.6	63.3	77.5
DeepSeek-R1-Distill-Qwen-7B	95.2	83.3	89.3

5.1. Can smaller policy models outperform larger models with the compute-optimal TTS strategy (Q4)

扩大小型策略模型的测试时计算对于提高大型语言模型（LLM）的推理性能至关重要。我们感兴趣的是，较小的策略模型是否能够通过计算最优的TTS策略胜过较大的模型，如GPT-4o、甚至o1和DeepSeek-R1。首先，我们比较了Llama-3.2-3B-Instruct（计算最优TTS）与Llama-3.1-405B-Instruct（CoT）在MATH-500和AIME24上的性能。此外，我们还比较了Qwen2.5-0.5B-Instruct、Qwen2.5-1.5B-Instruct、Llama-3.2-1B-Instruct和Llama-3.2-3B-Instruct与GPT-4o在上述两个任务上的性能。由于AIME24对当前的LLM来说具有挑战性，我们还比较了DeepSeek-R1-Distill-Qwen-1.5B和DeepSeek-R1-Distill-Qwen-7B与o1在AIME24上的性能。

从表 3的结果中，我们有以下观察：(1) 使用计算最优的TTS策略的Llama-3.2-3B-Instruct在MATH-500和AIME24上的表现优于Llama-3.1-405B-Instruct，这意味着较小的模型在使用计算最优的TTS策略时可以超过135倍大的模型。与之前关于TTS的工作 (Snell et al., 2024; Beeching et al., 2024)相比，我们将结果提高了487.0%（从23倍提高到135倍）。(2) 如果我们将计算预算进一步增加到 $N = 512$ ，使用计算最优的TTS策略的Llama-3.2-1B-Instruct在MATH-500上超过了Llama-3.1-405B-Instruct，

Table 2: Performance of TTS with different voting methods on MATH-500.

	Skywork-PRM-7B	Qwen2.5-Math-PRM-7B
<i>Majority Vote</i>	86.8	87.6
<i>PRM-Min-Max</i>	83.0	87.4
<i>PRM-Min-Vote</i>	86.6	87.6
<i>PRM-Last-Max</i>	84.4	87.6
<i>PRM-Last-Vote</i>	87.0	87.6
<i>PRM-Avg-Max</i>	85.8	87.8
<i>PRM-Avg-Vote</i>	86.8	87.6

errors in the search process.

5. Results for Compute-Optimal Test-Time Scaling

With the compute-optimal TTS strategy explored in Section 4, we conduct further experiments to explore the following questions:

- **Q4:** Can smaller policy models outperform larger models with the compute-optimal TTS strategy?
- **Q5:** How does compute-optimal TTS improve compared with CoT and majority voting?
- **Q6:** Is TTS more effective than long-CoT-based methods?

5.1. Can smaller policy models outperform larger models with the compute-optimal TTS strategy (Q4)

Scaling test-time compute of small policy models is crucially important for improving the reasoning performance of LLMs. We are interested in whether smaller policy models can outperform larger ones, GPT-4o, even o1 and DeepSeek-R1, with the compute-optimal TTS strategy. First, we compare the performance of Llama-3.2-3B-Instruct (compute-optimal TTS) with that of Llama-3.1-405B-Instruct (CoT) on MATH-500 and AIME24. Also, we compare the performance of Qwen2.5-0.5B-Instruct, Qwen2.5-1.5B-Instruct, Llama-3.2-1B-Instruct, and Llama-3.2-3B-Instruct with GPT-4o on the above two tasks. As AIME24 is challenging for current LLMs, we also compare the performance of DeepSeek-R1-Distill-Qwen-1.5B and DeepSeek-R1-Distill-Qwen-7B with o1 on AIME24.

From the results in Table 3, we have the following observations: (1) Llama-3.2-3B-Instruct with the compute-optimal TTS strategy outperforms Llama-3.1-405B-Instruct on MATH-500 and AIME24, meaning that **smaller models can outperform 135× larger models using the compute-optimal TTS strategy**. Compared with previous works on TTS (Snell et al., 2024; Beeching et al., 2024), we improve the result by **487.0%** ($23\times \rightarrow 135\times$). (2) If we further increase the compute budget to $N = 512$, Llama-3.2-1B-Instruct with the compute-optimal TTS strategy beats Llama-3.1-405B-Instruct on

Table 4: 较小策略模型（计算最优的TTS）与较大模型（CoT）之间的FLOPS比较。

Policy Model	Pre-training FLOPS	Inference FLOPS	Total FLOPS.
Llama-3.2-3B-Inst.	1.62×10^{23}	3.07×10^{17}	1.62×10^{23}
Llama-3.1-405B-Inst.	3.65×10^{25}	4.25×10^{17}	3.65×10^{25}
DeepSeek-R1-Distill-7B	7.56×10^{23}	8.15×10^{17}	7.56×10^{23}
DeepSeek-R1	5.96×10^{25}	4.03×10^{18}	5.96×10^{25}

但在AIME24上表现不如Llama-3.1-405B-Instruct。⁵ (3) 使用计算最优的TTS策略的Qwen2.5-0.5B-Instruct和Llama-3.2-3B-Instruct的表现优于GPT-4o，表明小型模型在使用计算最优的TTS策略时可以超过GPT级别的性能。(4) 使用计算最优的TTS策略的DeepSeek-R1-Distill-Qwen-1.5B在MATH-500和AIME24上的表现优于o1-preview和o1-mini。我们还展示了使用计算最优的TTS策略的DeepSeek-R1-Distill-Qwen-7B在MATH-500和AIME24上的表现优于o1和DeepSeek-R1。这些结果表明小型推理增强模型在使用计算最优的TTS策略时可以超过前沿推理LLM的表现。

FLOPS比较。 为了回答计算最优的TTS是否比增加模型大小更有效的问题，我们根据 Snell et al. (2024)在表 4中比较了评估模型的FLOPS，其中计算的FLOPS与表 3中的结果相对应。从结果中可以看出，小型策略模型即使在较少的推理FLOPS下也能超过大型模型，并且将总FLOPS减少了 $100\times \sim 1000\times$ 。

5.2. How does compute-optimal TTS improve compared with CoT and majority voting? (Q5)

基于不同策略模型、PRMs和难度级别的计算最优 TTS 的研究结果，我们在表 5 中总结了每个策略模型在 MATH-500 上的计算最优 TTS 结果。我们发现，计算最优 TTS 的效率可以比多数投票高 $256\times$ ，并且在 CoT 的基础上提高了 154.6% 的推理性能。这些结果表明，计算最优 TTS 显著增强了大语言模型的推理能力。然而，随着策略模型参数数量的增加，TTS 的改进逐渐减少。这表明 TTS 的有效性直接与策略模型的推理能力相关。具体来说，对于推理能力较弱的模型，增加测试时的计算量会导致显著的改进，而对于推理能力较强的模型，收益则有限。

5.3. Is TTS more effective than long-CoT-based methods? (Q6)

最近，基于长链思考（long-CoT）的方法在数学推理方面取得了显著进展 (Guan et al., 2025; Cui et al., 2025; Zeng et al., 2025; DeepSeek-AI et al., 2025)。我们将TTS与这些方法的性能进行比较。

设置。 我们评估了以下方法：(1) **rStar-Math** (Guan et al., 2025)：此方法首先通过MCTS生成推理数据，然后进行在线策略和偏好模型学习。(2) **Eurus-2** (Cui et al., 2025)：此方法通过隐式过

⁵由于Llama-3.2-1B-Instruct的一些输出中不包含用于答案提取的\boxed，我们使用Qwen2.5-32B-Instruct来提取Llama-3.2-1B-Instruct的答案。

Table 3: Comparison of small policy models (compute-optimal TTS) with frontier reasoning LLMs (CoT) on MATH-500 and AIME24.

Policy Model	MATH-500	AIME24	Avg.
<i>Proprietary LLMs (CoT)</i>			
GPT-4o	74.6	9.3	42.0
o1-preview	85.5	44.6	65.1
o1-mini	90.0	63.6	76.8
o1	94.8	79.2	87.0
<i>Open-Source LLMs (CoT)</i>			
Llama-3.1-70B-Inst.	65.2	16.7	41.0
Llama-3.1-405B-Inst.	71.4	23.3	47.4
QwQ-32B-Preview	90.6	50.0	70.3
DeepSeek-R1	97.3	79.8	88.6
<i>Open-Source LLMs (TTS)</i>			
Llama-3.2-1B-Inst.	66.2	16.7	41.5
Llama-3.2-1B-Inst. ($N = 512$)	72.2	10.0	41.1
Llama-3.2-3B-Inst.	75.6	30.0	52.8
Qwen2.5-0.5B-Inst.	76.4	10.0	43.2
Qwen2.5-1.5B-Inst.	81.8	20.0	50.9
DeepSeek-R1-Distill-Qwen-1.5B	91.6	63.3	77.5
DeepSeek-R1-Distill-Qwen-7B	95.2	83.3	89.3

MATH-500, but underperforms Llama-3.1-405B-Instruct on AIME24.⁵ (3) Qwen2.5-0.5B-Instruct and Llama-3.2-3B-Instruct with the compute-optimal TTS strategy outperforms GPT-4o, indicating that **small models can exceed GPT-level performance with the compute-optimal TTS strategy**. (4) DeepSeek-R1-Distill-Qwen-1.5B with the compute-optimal TTS strategy outperforms o1-preview and o1-mini on MATH-500 and AIME24. We also show that DeepSeek-R1-Distill-Qwen-7B with the compute-optimal TTS strategy outperforms o1 and DeepSeek-R1 on MATH-500 and AIME24. These results demonstrate **small reasoning-enhanced models can outperform frontier reasoning LLMs with the compute-optimal TTS strategy**.

FLOPS Comparison. To answer the question of whether compute-optimal TTS is more effective than increasing the model size, we compare the FLOPS of evaluated models in Table 4 following Snell et al. (2024), where the computed FLOPS is corresponded to the results in Table 3. From the results, we can see that **small policy models even surpass large ones with less inference FLOPS**

⁵Since some outputs of Llama-3.2-1B-Instruct do not contain \boxed, which is used for answer extraction, we use Qwen2.5-32B-Instruct to extract the answers of Llama-3.2-1B-Instruct.

Table 5: 在 MATH-500 上比较计算最优的 TTS、CoT 和不同策略模型下的多数投票。

Policy Model	CoT	Major.	Compute-Optimal TTS	Performance Gain	Efficiency Gain
Llama-3.2-1B-Inst.	26.0	39.0	66.2	154.6%	>256.0×
Llama-3.2-3B-Inst.	41.4	58.4	78.2	88.9%	14.1×
Llama-3.1-8B-Inst.	49.8	66.4	80.6	61.8%	43.9×
Qwen2.5-0.5B-Inst.	31.6	47.2	76.4	141.8%	>64.0×
Qwen2.5-1.5B-Inst.	54.4	68.4	85.6	57.4%	>256.0×
Qwen2.5-3B-Inst.	64.0	77.0	87.6	36.9%	58.4×
Qwen2.5-7B-Inst.	76.8	83.6	91.0	18.5%	35.9×
Qwen2.5-14B-Inst.	80.2	85.6	91.0	13.5%	51.4×
Qwen2.5-32B-Inst.	82.4	87.0	90.6	10.0%	0.8×
Qwen2.5-72B-Inst.	83.8	87.2	91.8	9.5%	12.9×

程奖励和在线强化学习（RL）增强大型语言模型（LLM）的推理能力。(3) **SimpleRL** (Zeng et al., 2025): 此方法仅使用8K训练数据复制自我反思。(4) **Satori** (Shen et al., 2025): 此方法首先学习格式, 然后通过RL提高推理能力。(5) **DeepSeek-R1-Distill-Qwen-7B** (DeepSeek-AI et al., 2025): 此方法从具有671B参数的DeepSeek-R1中提取800K高质量推理样本, 蒸馏到一个7B的LLM中。

结果。 如表 6 所示, 我们发现使用Qwen2.5-7B-Instruct的TTS在MATH-500和AIME24上都优于rStar-Math、Eurus-2、SimpleRL和Satori。然而, 尽管TTS在MATH-500上的表现接近DeepSeek-R1-Distill-Qwen-7B, 但在AIME24上表现显著下降。这些结果表明, TTS在通过MCTS生成的数据上应用直接强化学习（RL）或监督微调（SFT）的方法中更为有效, 但不如从强大的推理模型中蒸馏有效。此外, TTS在较简单的任务上比在更复杂的任务上更有效。

6. Related Work

LLM 测试时扩展。 扩展 LLM 测试时计算是一种有效提高性能的方法 (OpenAI, 2024)。先前的研究探索了多数投票 (Wang et al., 2023)、基于搜索的方法 (Yao et al., 2023; Xie et al., 2023; Khanov et al., 2024; Wan et al., 2024) 和改进 (Qu et al., 2024) 以提高性能。对于验证引导的测试时计算, Brown et al. (2024) 探索了通过重复采样和领域验证器进行推理计算, 而 Kang et al. (2024); Wu et al. (2024); Snell et al. (2024) 进一步探索了带有过程奖励指导的基于搜索的方法, Wang et al. (2024c) 将此设置扩展到了 VLMs。为了消除对外部奖励模型和生成大量样本的需求, Manvi et al. (2024) 提出了一种自评估方法, 用于适应性和高效的测试时计算。最近的一项工作 (Beeching et al., 2024) 通过具有多样性的搜索方法探索了 TTS。然而, 这些工作缺乏使用强大的验证器或不同规模/能力的策略的评估。在本文中, 我们旨在通过使用最新的策略和验证器、更具挑战性的任务, 提供更系统的评估, 并提供一些实用的 TTS 原则。

提高 LLM 的数学推理能力。 提高数学推理能力的先前方法可以分为训练时方法和测试时方法。对于训练时方法, 先前的研究探索了大规模数学语料库预训练 (OpenAI, 2023; Azerbayev et al., 2024;

Table 4: FLOPS comparison between smaller policy models (compute-optimal TTS) and larger ones (CoT).

Policy Model	Pre-training FLOPS	Inference FLOPS	Total FLOPS.
Llama-3.2-3B-Inst.	1.62×10^{23}	3.07×10^{17}	1.62×10^{23}
Llama-3.1-405B-Inst.	3.65×10^{25}	4.25×10^{17}	3.65×10^{25}
DeepSeek-R1-Distill-7B	7.56×10^{23}	8.15×10^{17}	7.56×10^{23}
DeepSeek-R1	5.96×10^{25}	4.03×10^{18}	5.96×10^{25}

Table 5: Comparison of compute-optimal TTS, CoT, and majority voting with different policy models on MATH-500.

Policy Model	CoT	Major.	Compute-Optimal TTS	Performance Gain	Efficiency Gain
Llama-3.2-1B-Inst.	26.0	39.0	66.2	154.6%	$>256.0\times$
Llama-3.2-3B-Inst.	41.4	58.4	78.2	88.9%	$14.1\times$
Llama-3.1-8B-Inst.	49.8	66.4	80.6	61.8%	$43.9\times$
Qwen2.5-0.5B-Inst.	31.6	47.2	76.4	141.8%	$>64.0\times$
Qwen2.5-1.5B-Inst.	54.4	68.4	85.6	57.4%	$>256.0\times$
Qwen2.5-3B-Inst.	64.0	77.0	87.6	36.9%	$58.4\times$
Qwen2.5-7B-Inst.	76.8	83.6	91.0	18.5%	$35.9\times$
Qwen2.5-14B-Inst.	80.2	85.6	91.0	13.5%	$51.4\times$
Qwen2.5-32B-Inst.	82.4	87.0	90.6	10.0%	$0.8\times$
Qwen2.5-72B-Inst.	83.8	87.2	91.8	9.5%	$12.9\times$

and reduce the total FLOPS by $100\times \sim 1000\times$.

5.2. How does compute-optimal TTS improve compared with CoT and majority voting? (Q5)

Based on the findings of compute-optimal TTS with different policy models, PRMs, and difficulty levels, we summarize the results of compute-optimal TTS for each policy model on MATH-500 in Table 5. We find that compute-optimal TTS can be $256\times$ more efficient than majority voting and improve reasoning performance by 154.6% over CoT. These results demonstrate that compute-optimal TTS significantly enhances the reasoning capabilities of LLMs. However, as the number of parameters in the policy model increases, the improvement of TTS gradually decreases. This suggests that the effectiveness of TTS is directly related to the reasoning ability of the policy model. Specifically, for models with weak reasoning abilities, scaling test-time compute leads to a substantial improvement, whereas for models with strong reasoning abilities, the gain is limited.

Table 6: 计算最优的TTS与长链思考方法在MATH-500和AIME24上的比较。

Policy Model	MATH-500	AIME24	Avg.
<i>Open-Source LLMs (CoT)</i>			
Qwen2.5-7B-Inst.	76.8	13.3	45.1
Qwen2.5-Math-7B-Inst.	79.8	13.3	46.6
<i>Long-CoT Methods (CoT)</i>			
rStar-Math-7B	78.4	26.7	52.6
Eurus-2-7B-PRIME	79.2	26.7	53.0
Qwen2.5-7B-SimpleRL-Zero	77.2	33.3	55.3
Qwen2.5-7B-SimpleRL	82.4	26.7	54.6
Satori-Qwen-7B	83.6	23.3	53.5
DeepSeek-R1-Distill-Qwen-7B	92.4	63.3	77.9
<i>Open-Source LLMs (TTS)</i>			
Qwen2.5-7B-Inst. w/ 7B PRM (Ours)	88.0	33.3	60.5
Qwen2.5-7B-Inst. w/ 72B PRM (Ours)	91.0	36.7	63.9

Shao et al., 2024) 和监督微调 (Luo et al., 2023; Yu et al., 2024; Gou et al., 2024; Tang et al., 2024; Tong et al., 2024; Zeng et al., 2024) 以提高数学能力。另一系列的工作探索了自训练和自改进策略 (Zelikman et al., 2022; Gulcehre et al., 2023; Trung et al., 2024; Hosseini et al., 2024; Zelikman et al., 2024; Zhang et al., 2024a; Setlur et al., 2024a; Kumar et al., 2024; Cui et al., 2025), 通过在自生成的解决方案上进行微调来提高推理能力。最近, 许多工作通过长 CoT (Qin et al., 2024; Huang et al., 2024; Kimi, 2024; DeepSeek-AI et al., 2025; Qwen Team, 2024; Skywork, 2024; Zhao et al., 2024) 改进了数学推理能力, 正如 OpenAI o1 (OpenAI, 2024) 所展示的, 长思考显著提高了推理能力。

对于测试时方法, 基于提示的方法已被广泛研究, 以在不改变模型参数的情况下增强推理能力。诸如 Chain-of-Thought (CoT) (Wei et al., 2022) 及其变体 (Yao et al., 2023; Leang et al., 2024) 等技术指导模型将问题分解为可管理的子步骤, 从而提高数学推理的准确性和连贯性。除了提示策略之外, 自改进技术 (Madaan et al., 2023) 允许模型审查和纠正其输出, 而外部工具集成 (Gao et al., 2023; Chen et al., 2023) 则利用程序解释器或符号操作器进行精确计算和验证。自验证方法 (Weng et al., 2023) 使模型能够评估其推理过程的正确性, 进一步提高鲁棒性。这些测试时策略与训练时增强相辅相成, 共同显著提高了 LLMs 的数学推理能力。我们的工作主要通过 PRM 引导的搜索方法扩展测试时计算来提高推理性能。

过程奖励模型。 先前的研究表明, PRMs比 ORMs更有效 (Uesato et al., 2022; Lightman et al., 2024)。然而, 收集高质量的 PRMs数据, 如 PRM800K (Lightman et al., 2024), 通常成本很高。研究人员探索了通过直接蒙特卡洛估计自动收集 PRM数据的方法 (Wang et al., 2024b), 检测 ORMs的相对得分 (Lu et al., 2024), 以及使用二分搜索的高效 MCTS (Luo et al., 2024)。最近, 从优势建模 (Setlur et al., 2024b)、Q-值排名 (Li and Li, 2024)、隐式奖励 (Yuan et al., 2024) 和熵正则化 (Zhang et al.,

Table 6: Comparison of compute-optimal TTS with long-CoT methods on MATH-500 and AIME24.

Policy Model	MATH-500	AIME24	Avg.
<i>Open-Source LLMs (CoT)</i>			
Qwen2.5-7B-Inst.	76.8	13.3	45.1
Qwen2.5-Math-7B-Inst.	79.8	13.3	46.6
<i>Long-CoT Methods (CoT)</i>			
rStar-Math-7B	78.4	26.7	52.6
Eurus-2-7B-PRIME	79.2	26.7	53.0
Qwen2.5-7B-SimpleRL-Zero	77.2	33.3	55.3
Qwen2.5-7B-SimpleRL	82.4	26.7	54.6
Satori-Qwen-7B	83.6	23.3	53.5
DeepSeek-R1-Distill-Qwen-7B	92.4	63.3	77.9
<i>Open-Source LLMs (TTS)</i>			
Qwen2.5-7B-Inst. w/ 7B PRM (Ours)	88.0	33.3	60.5
Qwen2.5-7B-Inst. w/ 72B PRM (Ours)	91.0	36.7	63.9

5.3. Is TTS more effective than long-CoT-based methods? (Q6)

Recently, long-CoT-based methods have shown substantial progress in mathematical reasoning (Guan et al., 2025; Cui et al., 2025; Zeng et al., 2025; DeepSeek-AI et al., 2025). We compare the performance of TTS with these approaches.

Setup. We evaluate the following methods: (1) **rStar-Math** (Guan et al., 2025): This method first generates reasoning data via MCTS, followed by online policy and preference model learning. (2) **Eurus-2** (Cui et al., 2025): This method enhances the reasoning abilities of LLMs through implicit process rewards and online RL. (3) **SimpleRL** (Zeng et al., 2025): This method replicates self-reflection with only 8K training data. (4) **Satori** (Shen et al., 2025): This method first learn the format and then improves the reasoning abilities via RL. (5) **DeepSeek-R1-Distill-Qwen-7B** (DeepSeek-AI et al., 2025): This method distills 800K high-quality reasoning samples from DeepSeek-R1 with 671B parameters into a 7B LLM.

Results. As shown in Table 6, we find that TTS with Qwen2.5-7B-Instruct outperforms rStar-Math, Eurus-2, SimpleRL, and Satori on both MATH-500 and AIME24. However, while the performance of TTS on MATH-500 is close to that of DeepSeek-R1-Distill-Qwen-7B, it shows a significant drop on AIME24. These results indicate that TTS is more effective than methods applying direct RL or SFT on the data generated via MCTS but is less effective than distilling from strong reasoning models. Also, TTS is more effective on simpler tasks than on more complex tasks.

2024b) 等角度探索了更先进的 PRMs。此外，更多开源的 PRMs 被发布 (Xiong et al., 2024; Skywork, 2024; Zhang et al., 2024b; Li and Li, 2024; Yuan et al., 2024; Zhang et al., 2025)，在数学任务上表现出强大的性能。随着 PRMs 的快速发展，提出了 ProcessBench (Zheng et al., 2024) 和 PRMBench (Song et al., 2025) 以提供对 PRMs 的全面评估。Zhang et al. (2025) 提供了 PRMs 实际开发的指导方针，并发布了迄今为止最强大的用于数学任务的 PRMs。

7. Conclusion & Discussion

在本文中，我们从不同策略模型、PRMs 和更具挑战性的评估任务的角度，对计算最优的测试时间扩展 (TTS) 进行了全面的实证分析。我们的研究表明，计算最优的 TTS 策略依赖于策略模型、PRMs 和问题难度，验证了在应用计算最优 TTS 时，较小的语言模型可以比大型模型表现得更好。我们的结果表明，通过 TTS，1B 的模型可以比 405B 的模型表现得更好。此外，我们还证明了 7B 的 PRM 通过监督一个更强大的 72B 策略模型可以实现强大的 TTS 结果，这表明研究真正的“弱到强”方法而不是当前的“强到弱”监督对于策略优化的重要性。为了实现这一目标，我们需要开发更有效的监督方法，因为基于 PRM 和基于 RL 的方法都由于依赖高质量的监督而存在局限性。未来的工作应集中在开发更适应性和普遍性的监督机制，以提高小型语言模型在复杂任务上的性能，并为开发高效的推理策略提供新的方法。

局限性。 尽管我们对数学任务上的 TTS 进行了全面的评估，但仍有一些局限性和未来的研究方向需要探索：(1) 将 TTS 扩展到更多任务，如编程和化学任务。(2) 探索更有效的计算最优 TTS 方法。

6. Related Work

LLM Test-Time Scaling. Scaling LLM test-time compute is an effective way to improve the performance (OpenAI, 2024). Previous works explore majority voting (Wang et al., 2023), search-based methods (Yao et al., 2023; Xie et al., 2023; Khanov et al., 2024; Wan et al., 2024), and refinement (Qu et al., 2024) to improve the performance. For verification-guided test-time compute, Brown et al. (2024) explores inference compute with repeated sampling and domain verifiers, while Kang et al. (2024); Wu et al. (2024); Snell et al. (2024) further explore search-based methods with process reward guidance and Wang et al. (2024c) extends this setting to VLMs. To eliminate the need for external reward models and the generation of extensive samples, Manvi et al. (2024) proposes a self-evaluation method for adaptive and efficient test-time compute. A recent work (Beeching et al., 2024) explores TTS via search methods with diversity. However, these works lack a evaluation with either strong verifiers or policies with different sizes / capabilities. In this paper, we aim to provide a more systematically evaluation with up-to-date policies and verifiers, more challenging tasks, and provide some principles for practical TTS.

Improving Mathematical Reasoning Abilities of LLMs. Prior methods for improving mathematical reasoning abilities can be divided into training-time methods and test-time methods. For training-time methods, previous works explore large-scale mathematical corpus pre-training (OpenAI, 2023; Azerbayev et al., 2024; Shao et al., 2024) and supervised fine-tuning (Luo et al., 2023; Yu et al., 2024; Gou et al., 2024; Tang et al., 2024; Tong et al., 2024; Zeng et al., 2024) to improve mathematical capabilities. Another line of works explore self-training and self-improvement strategies (Zelikman et al., 2022; Gulcehre et al., 2023; Trung et al., 2024; Hosseini et al., 2024; Zelikman et al., 2024; Zhang et al., 2024a; Setlur et al., 2024a; Kumar et al., 2024; Cui et al., 2025), which improve the reasoning abilities by fine-tuning on self-generated solutions. Recently, many works improve the mathematical reasoning abilities with long CoT (Qin et al., 2024; Huang et al., 2024; Kimi, 2024; DeepSeek-AI et al., 2025; Qwen Team, 2024; Skywork, 2024; Zhao et al., 2024), as OpenAI o1 (OpenAI, 2024) shows significantly powerful reasoning capabilities with long thinking.

For test-time methods, prompt-based approaches have been extensively studied to enhance reasoning without altering the model parameters. Techniques such as Chain-of-Thought (CoT) (Wei et al., 2022) and its variants (Yao et al., 2023; Leang et al., 2024) guide the model to decompose problems into manageable sub-steps, thereby improving accuracy and coherence in mathematical reasoning. Beyond prompting strategies, self-refinement techniques (Madaan et al., 2023) allow models to review and correct their outputs, while external tool integration (Gao et al., 2023; Chen et al., 2023) leverages program interpreter or symbolic manipulators to perform precise calculations and validations. Self-verification approaches (Weng et al., 2023) enable models to assess the correctness of their own reasoning processes, further increasing robustness. These test-time strategies complement training-time enhancements, collectively contributing to significant improvements in LLMs’ mathematical

References

- AI-MO. Aime 2024, 2024. URL <https://huggingface.co/datasets/AI-MO/aimo-validation-aime>.
- Anthropic. Introducing Claude, 2023. URL <https://www.anthropic.com/index/introducing-claude/>.
- Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen Marcus McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics. In *International Conference on Learning Representations (ICLR)*, 2024. URL <https://openreview.net/forum?id=4WnqRR915j>.
- Edward Beeching, Lewis Tunstall, and Sasha Rush. Scaling test-time compute with open models, 2024. URL <https://huggingface.co/spaces/HuggingFaceH4/blogpost-scaling-test-time-compute>.
- Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.
- Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. Alphamath almost zero: Process supervision without process. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL <https://openreview.net/forum?id=VaXnxQ3UKo>.
- Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research (TMLR)*, 2023. ISSN 2835-8856. URL <https://openreview.net/forum?id=YfZ4ZPt8zd>.
- Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.
- 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,

reasoning capabilities. Our work mainly enhances the reasoning performance via scaling test-time compute via PRM-guided search methods.

Process Reward Models. Previous works show that PRMs are more effective than ORMs (Uesato et al., 2022; Lightman et al., 2024). However, collecting high-quality PRMs data, such as PRM800K (Lightman et al., 2024), is often costly. The researchers explores automatic PRM data collection via direct Monte Carlo estimation (Wang et al., 2024b), detecting relative scores of ORMs (Lu et al., 2024), and efficient MCTS with binary search (Luo et al., 2024). Recently, more advanced PRMs are explored from advantage modeling (Setlur et al., 2024b), Q -value rankings (Li and Li, 2024), implicit rewards (Yuan et al., 2024), and entropy regularization (Zhang et al., 2024b) perspectives. Additionally, more open-source PRMs are released (Xiong et al., 2024; Skywork, 2024; Zhang et al., 2024b; Li and Li, 2024; Yuan et al., 2024; Zhang et al., 2025), showing strong performance on mathematical tasks. With the rapid development of PRMs, ProcessBench (Zheng et al., 2024) and PRMBench (Song et al., 2025) are proposed to provide comprehensive evaluation of PRMs. Zhang et al. (2025) provides guidelines for practical development of PRMs and releases the most capable PRMs for mathematical tasks up-to-date.

7. Conclusion & Discussion

In this paper, we present a thorough empirical analysis of compute-optimal test-time scaling from the perspectives of different policy models, PRMs, and more challenging evaluation tasks. Our findings demonstrate the dependency of compute-optimal TTS strategies on policy models, PRMs, and problem difficulty, validating that smaller language models can perform better than larger models when applying compute-optimal TTS. Our results show that a 1B model can achieve better performance than a 405B model through TTS. Additionally, we demonstrate that a 7B PRM can achieve strong TTS results by supervising a more capable 72B policy model, which suggests the importance of investigating a true “weak-to-strong” approach instead of the current “strong-to-weak” supervision for policy optimization. To achieve this goal, we need to develop more efficient supervision methods, as both PRM-based and RL-based approaches have limitations due to their dependence on high-quality supervision. Future work should focus on developing more adaptable and universal supervision mechanisms to boost the performance of small language models on complex tasks and provide new approaches for developing efficient reasoning strategies.

Limitations. Although we provide a comprehensive evaluation of TTS on mathematical tasks, there are still some limitations and future directions to explore: (1) Extending TTS to more tasks such as coding and chemistry tasks. (2) Exploring more effective methods for compute-optimal TTS.

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, Wanxia 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, Yudian 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. *arXiv preprint arXiv:2501.12948*, 2025.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *International Conference on Machine Learning (ICML)*, volume 202, pages 10764–10799, 2023.

Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. ToRA: A tool-integrated reasoning agent for mathematical problem solving. In *International Conference on Learning Representations (ICLR)*, 2024. URL <https://openreview.net/forum?id=Ep0TtjVoap>.

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

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

References

- AI-MO. Aime 2024, 2024. URL <https://huggingface.co/datasets/AI-MO/aimo-validation-aime>.
- Anthropic. Introducing Claude, 2023. URL <https://www.anthropic.com/index/introducing-claude/>.
- Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen Marcus McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics. In *International Conference on Learning Representations (ICLR)*, 2024. URL <https://openreview.net/forum?id=4WnqRR915j>.
- Edward Beeching, Lewis Tunstall, and Sasha Rush. Scaling test-time compute with open models, 2024. URL <https://huggingface.co/spaces/HuggingFaceH4/blogpost-scaling-test-time-compute>.
- Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.
- Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. Alphamath almost zero: Process supervision without process. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL <https://openreview.net/forum?id=VaXnxQ3UKo>.
- Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research (TMLR)*, 2023. ISSN 2835-8856. URL <https://openreview.net/forum?id=YfZ4ZPt8zd>.
- Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.
- 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,
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL <https://openreview.net/forum?id=7Bywt2mQsCe>.
- Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. V-star: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*, 2024.
- Zhen Huang, Haoyang Zou, Xuefeng Li, Yixiu Liu, Yuxiang Zheng, Ethan Chern, Shijie Xia, Yiwei Qin, Weizhe Yuan, and Pengfei Liu. O1 replication journey—part 2: Surpassing o1-preview through simple distillation, big progress or bitter lesson? *arXiv preprint arXiv:2411.16489*, 2024.
- Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.
- Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.
- Jikun Kang, Xin Zhe Li, Xi Chen, Amirreza Kazemi, Qianyi Sun, Boxing Chen, Dong Li, Xu He, Quan He, Feng Wen, et al. MindStar: Enhancing math reasoning in pre-trained llms at inference time. *arXiv preprint arXiv:2405.16265*, 2024.
- Maxim Khanov, Jirayu Burapachee, and Yixuan Li. ARGS: Alignment as reward-guided search. In *International Conference on Learning Representations (ICLR)*, 2024. URL <https://openreview.net/forum?id=shgx0eqdw6>.
- Kimi. k0-math, November 2024. URL <https://kimi.moonshot.cn/>.
- Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1.5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.
- Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*, 2024.
- Joshua Ong Jun Leang, Aryo Pradipta Gema, and Shay B Cohen. CoMAT: Chain of mathematically annotated thought improves mathematical reasoning. *arXiv preprint arXiv:2410.10336*, 2024.
- Wendi Li and Yixuan Li. Process reward model with q-value rankings. *arXiv preprint arXiv:2410.11287*, 2024.

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, Wanxia 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. *arXiv preprint arXiv:2501.12948*, 2025.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *International Conference on Machine Learning (ICML)*, volume 202, pages 10764–10799, 2023.

Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. ToRA: A tool-integrated reasoning agent for mathematical problem solving. In *International Conference on Learning Representations (ICLR)*, 2024. URL <https://openreview.net/forum?id=Ep0TtjVoap>.

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

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

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 *International Conference on Learning Representations (ICLR)*, 2024. URL <https://openreview.net/forum?id=v8L0pN6E0i>.

Jianqiao Lu, Zhiyang Dou, WANG Hongru, Zeyu Cao, Jianbo Dai, Yunlong Feng, and Zhijiang Guo. Autopsv: Automated process-supervised verifier. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023.

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

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 46534–46594, 2023.

Rohin Manvi, Anikait Singh, and Stefano Ermon. Adaptive inference-time compute: Llms can predict if they can do better, even mid-generation. *arXiv preprint arXiv:2410.02725*, 2024.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

OpenAI. Learning to reason with llms, 2024. URL <https://openai.com/index/learning-to-reason-with-llms/>.

Yiwei Qin, Xuefeng Li, Haoyang Zou, Yixiu Liu, Shijie Xia, Zhen Huang, Yixin Ye, Weizhe Yuan, Hector Liu, Yuanzhi Li, et al. O1 replication journey: A strategic progress report–part 1. *arXiv preprint arXiv:2410.18982*, 2024.

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching language model agents how to self-improve. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL <https://openreview.net/forum?id=DRC9pZwBwR>.

Qwen Team. Qwq: Reflect deeply on the boundaries of the unknown, November 2024. URL <https://qwenlm.github.io/blog/qwq-32b-preview/>.

Amrith Setlur, Saurabh Garg, Xinyang Geng, Naman Garg, Virginia Smith, and Aviral Kumar. RL on incorrect synthetic data scales the efficiency of llm math reasoning by eight-fold. *arXiv preprint arXiv:2406.14532*, 2024a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL <https://openreview.net/forum?id=7Bywt2mQsCe>.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. V-star: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*, 2024.

Zhen Huang, Haoyang Zou, Xuefeng Li, Yixiu Liu, Yuxiang Zheng, Ethan Chern, Shijie Xia, Yiwei Qin, Weizhe Yuan, and Pengfei Liu. O1 replication journey—part 2: Surpassing o1-preview through simple distillation, big progress or bitter lesson? *arXiv preprint arXiv:2411.16489*, 2024.

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

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

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

Maxim Khanov, Jirayu Burapachee, and Yixuan Li. ARGS: Alignment as reward-guided search. In *International Conference on Learning Representations (ICLR)*, 2024. URL <https://openreview.net/forum?id=shgx0eqdw6>.

Kimi. k0-math, November 2024. URL <https://kimi.moonshot.cn/>.

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

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*, 2024.

Joshua Ong Jun Leang, Aryo Pradipta Gema, and Shay B Cohen. CoMAT: Chain of mathematically annotated thought improves mathematical reasoning. *arXiv preprint arXiv:2410.10336*, 2024.

Wendi Li and Yixuan Li. Process reward model with q-value rankings. *arXiv preprint arXiv:2410.11287*, 2024.

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

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

Maohao Shen, Guangtao Zeng, Zhenting Qi, Zhang-Wei Hong, Zhenfang Chen, Wei Lu, Gregory Wornell, Subhro Das, David Cox, and Chuang Gan. Satori: Reinforcement learning with chain-of-action-thought enhances llm reasoning via autoregressive search. *arXiv preprint arXiv:2502.02508*, 2025.

Skywork. Skywork-o1, November 2024. URL <https://www.tiangong.cn/>.

Skywork o1 Team. Skywork-o1 open series. <https://huggingface.co/Skywork>, November 2024. URL <https://huggingface.co/Skywork>.

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

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. Prmbench: A fine-grained and challenging benchmark for process-level reward models. *arXiv preprint arXiv:2501.03124*, 2025.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Zhengyang Tang, Xingxing Zhang, Benyou Wang, and Furu Wei. MathScale: Scaling instruction tuning for mathematical reasoning. In *International Conference on Machine Learning (ICML)*, volume 235, pages 47885–47900, 2024.

Yuxuan Tong, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. DART-math: Difficulty-aware rejection tuning for mathematical problem-solving. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL <https://openreview.net/forum?id=zLU21oQjD5>.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Luong Trung, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. Left: Reasoning with reinforced fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7601–7614, 2024.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

- 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 *International Conference on Learning Representations (ICLR)*, 2024. URL <https://openreview.net/forum?id=v8L0pN6E0i>.
- Jianqiao Lu, Zhiyang Dou, WANG Hongru, Zeyu Cao, Jianbo Dai, Yunlong Feng, and Zhijiang Guo. Autopsv: Automated process-supervised verifier. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.
- Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023.
- Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, et al. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 46534–46594, 2023.
- Rohin Manvi, Anikait Singh, and Stefano Ermon. Adaptive inference-time compute: LLMs can predict if they can do better, even mid-generation. *arXiv preprint arXiv:2410.02725*, 2024.
- OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- OpenAI. Learning to reason with llms, 2024. URL <https://openai.com/index/learning-to-reason-with-llms/>.
- Yiwei Qin, Xuefeng Li, Haoyang Zou, Yixiu Liu, Shijie Xia, Zhen Huang, Yixin Ye, Weizhe Yuan, Hector Liu, Yuanzhi Li, et al. O1 replication journey: A strategic progress report—part 1. *arXiv preprint arXiv:2410.18982*, 2024.
- Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching language model agents how to self-improve. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL <https://openreview.net/forum?id=DRC9pZwBwR>.
- Qwen Team. Qwq: Reflect deeply on the boundaries of the unknown, November 2024. URL <https://qwenlm.github.io/blog/qwq-32b-preview/>.
- Amrith Setlur, Saurabh Garg, Xinyang Geng, Naman Garg, Virginia Smith, and Aviral Kumar. RL on incorrect synthetic data scales the efficiency of llm math reasoning by eight-fold. *arXiv preprint arXiv:2406.14532*, 2024a.

- Ziyu Wan, Xidong Feng, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. AlphaZero-like tree-search can guide large language model decoding and training. In *International Conference on Machine Learning (ICML)*, volume 235, pages 49890–49920, 2024.
- Jun Wang, Meng Fang, Ziyu Wan, Muning Wen, Jiachen Zhu, Anjie Liu, Ziqin Gong, Yan Song, Lei Chen, Lionel M Ni, et al. Openr: An open source framework for advanced reasoning with large language models. *arXiv preprint arXiv:2410.09671*, 2024a.
- Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, 2024b.
- Xiyao Wang, Zhengyuan Yang, Linjie Li, Hongjin Lu, Yuancheng Xu, Chung-Ching Lin, Kevin Lin, Furong Huang, and Lijuan Wang. Scaling inference-time search with vision value model for improved visual comprehension. *arXiv preprint arXiv:2412.03704*, 2024c.
- 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 *International Conference on Learning Representations (ICLR)*, 2023. URL <https://openreview.net/forum?id=1PL1NIMMrw>.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in neural information processing systems (NeurIPS)*, volume 35, pages 24824–24837, 2022.
- Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2550–2575, 2023.
- Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*, 2024.
- Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. Self-evaluation guided beam search for reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 41618–41650, 2023.
- Wei Xiong, Hanning Zhang, Nan Jiang, and Tong Zhang. An implementation of generative prm. <https://github.com/RLHFlow/RLHF-Reward-Modeling>, 2024.
- An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang,

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

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

Maohao Shen, Guangtao Zeng, Zhenting Qi, Zhang-Wei Hong, Zhenfang Chen, Wei Lu, Gregory Wornell, Subhro Das, David Cox, and Chuang Gan. Satori: Reinforcement learning with chain-of-action-thought enhances llm reasoning via autoregressive search. *arXiv preprint arXiv:2502.02508*, 2025.

Skywork. Skywork-o1, November 2024. URL <https://www.tiangong.cn/>.

Skywork o1 Team. Skywork-o1 open series. <https://huggingface.co/Skywork>, November 2024. URL <https://huggingface.co/Skywork>.

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

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. Prmbench: A fine-grained and challenging benchmark for process-level reward models. *arXiv preprint arXiv:2501.03124*, 2025.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Zhengyang Tang, Xingxing Zhang, Benyou Wang, and Furu Wei. MathScale: Scaling instruction tuning for mathematical reasoning. In *International Conference on Machine Learning (ICML)*, volume 235, pages 47885–47900, 2024.

Yuxuan Tong, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. DART-math: Difficulty-aware rejection tuning for mathematical problem-solving. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL <https://openreview.net/forum?id=zLU21oQjD5>.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Luong Trung, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. Reft: Reasoning with reinforced fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7601–7614, 2024.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024a.

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, Jiayi 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, 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. *arXiv preprint arXiv:2412.15115*, 2024b.

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

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 11809–11822, 2023.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng YU, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap your own mathematical questions for large language models. In *International Conference on Learning Representations (ICLR)*, 2024. URL <https://openreview.net/forum?id=N8N0hgNDRt>.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels. *arXiv preprint arXiv:2412.01981*, 2024.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 15476–15488, 2022.

Eric Zelikman, Georges Raif Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah Goodman. Quiet-STaR: Language models can teach themselves to think before speaking. In *Conference on Language Modeling (COLM)*, 2024. URL <https://openreview.net/forum?id=oRXPiSOGH9>.

- Ziyu Wan, Xidong Feng, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. AlphaZero-like tree-search can guide large language model decoding and training. In *International Conference on Machine Learning (ICML)*, volume 235, pages 49890–49920, 2024.
- Jun Wang, Meng Fang, Ziyu Wan, Muning Wen, Jiachen Zhu, Anjie Liu, Ziqin Gong, Yan Song, Lei Chen, Lionel M Ni, et al. Openr: An open source framework for advanced reasoning with large language models. *arXiv preprint arXiv:2410.09671*, 2024a.
- Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, 2024b.
- Xiyao Wang, Zhengyuan Yang, Linjie Li, Hongjin Lu, Yuancheng Xu, Chung-Ching Lin, Kevin Lin, Furong Huang, and Lijuan Wang. Scaling inference-time search with vision value model for improved visual comprehension. *arXiv preprint arXiv:2412.03704*, 2024c.
- 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 *International Conference on Learning Representations (ICLR)*, 2023. URL <https://openreview.net/forum?id=1PL1NIMMrw>.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in neural information processing systems (NeurIPS)*, volume 35, pages 24824–24837, 2022.
- Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2550–2575, 2023.
- Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*, 2024.
- Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. Self-evaluation guided beam search for reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 41618–41650, 2023.
- Wei Xiong, Hanning Zhang, Nan Jiang, and Tong Zhang. An implementation of generative prm. <https://github.com/RLHFlow/RLHF-Reward-Modeling>, 2024.
- An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang,

- Liang Zeng, Liangjun Zhong, Liang Zhao, Tianwen Wei, Liu Yang, Jujie He, Cheng Cheng, Rui Hu, Yang Liu, Shuicheng Yan, et al. Skywork-Math: Data scaling laws for mathematical reasoning in large language models—the story goes on. *arXiv preprint arXiv:2407.08348*, 2024.
- Weihaio Zeng, Yuzhen Huang, Wei Liu, Keqing He, Qian Liu, Zejun Ma, and Junxian He. 7b model and 8k examples: Emerging reasoning with reinforcement learning is both effective and efficient. <https://hkust-nlp.notion.site/simpler1-reason>, 2025. Notion Blog.
- Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. ReST-MCTS*: LLM self-training via process reward guided tree search. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a. URL <https://openreview.net/forum?id=8rcF0qEud5>.
- Hanning Zhang, Pengcheng Wang, Shizhe Diao, Yong Lin, Rui Pan, Hanze Dong, Dylan Zhang, Pavlo Molchanov, and Tong Zhang. Entropy-regularized process reward model. *arXiv preprint arXiv:2412.11006*, 2024b.
- 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. *arXiv preprint arXiv:2501.07301*, 2025.
- Yu Zhao, Huifeng Yin, Bo Zeng, Hao Wang, Tianqi Shi, Chenyang Lyu, Longyue Wang, Weihua Luo, and Kaifu Zhang. Marco-o1: Towards open reasoning models for open-ended solutions. *arXiv preprint arXiv:2411.14405*, 2024.
- Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Processbench: Identifying process errors in mathematical reasoning. *arXiv preprint arXiv:2412.06559*, 2024.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 46595–46623, 2023.

Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024a.

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, Jiayi 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, 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. *arXiv preprint arXiv:2412.15115*, 2024b.

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

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 11809–11822, 2023.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng YU, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap your own mathematical questions for large language models. In *International Conference on Learning Representations (ICLR)*, 2024. URL <https://openreview.net/forum?id=N8N0hgNDRt>.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels. *arXiv preprint arXiv:2412.01981*, 2024.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 15476–15488, 2022.

Eric Zelikman, Georges Raif Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah Goodman. Quiet-STaR: Language models can teach themselves to think before speaking. In *Conference on Language Modeling (COLM)*, 2024. URL <https://openreview.net/forum?id=oRXPiSOGH9>.

A. Prompt Template for Test-Time Scaling

Llama 3 系列模型 (Dubey et al., 2024) 和 Qwen2.5 系列模型 (Yang et al., 2024b) 的系统提示分别列在表 7 和表 8 中。根据 Beeching et al. (2024), 我们使用 Llama 3 的官方评估系统的提示⁶, 以防止性能下降。

Table 7: System prompt for Llama 3 series models.

<div>Solve the following math problem efficiently and clearly:</div> <div><div>- For simple problems (2 steps or fewer): Provide a concise solution with minimal explanation.</div><div>- For complex problems (3 steps or more): Use this step-by-step format:</div></div> <div>## Step 1: [Concise description] [Brief explanation and calculations]</div> <div>## Step 2: [Concise description] [Brief explanation and calculations]</div> <div>...</div> <div>Regardless of the approach, always conclude with:</div> <div>Therefore, the final answer is: $\boxed{\text{answer}}$. I hope it is correct.</div> <div>Where [answer] is just the final number or expression that solves the problem.</div>
--

Table 8: System prompt for Qwen2.5 series models.

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

⁶<https://huggingface.co/datasets/meta-llama/Llama-3.2-1B-Instruct-evals>

Liang Zeng, Liangjun Zhong, Liang Zhao, Tianwen Wei, Liu Yang, Jujie He, Cheng Cheng, Rui Hu, Yang Liu, Shuicheng Yan, et al. Skywork-Math: Data scaling laws for mathematical reasoning in large language models—the story goes on. *arXiv preprint arXiv:2407.08348*, 2024.

Weihao Zeng, Yuzhen Huang, Wei Liu, Keqing He, Qian Liu, Zejun Ma, and Junxian He. 7b model and 8k examples: Emerging reasoning with reinforcement learning is both effective and efficient. <https://hkust-nlp.notion.site/simpler1-reason>, 2025. Notion Blog.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. ReST-MCTS*: LLM self-training via process reward guided tree search. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a. URL <https://openreview.net/forum?id=8rcF0qEud5>.

Hanning Zhang, Pengcheng Wang, Shizhe Diao, Yong Lin, Rui Pan, Hanze Dong, Dylan Zhang, Pavlo Molchanov, and Tong Zhang. Entropy-regularized process reward model. *arXiv preprint arXiv:2412.11006*, 2024b.

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. *arXiv preprint arXiv:2501.07301*, 2025.

Yu Zhao, Huifeng Yin, Bo Zeng, Hao Wang, Tianqi Shi, Chenyang Lyu, Longyue Wang, Weihua Luo, and Kaifu Zhang. Marco-o1: Towards open reasoning models for open-ended solutions. *arXiv preprint arXiv:2411.14405*, 2024.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 46595–46623, 2023.

B. Full Results of Test-Time Scaling with Different Policy Models, PRMs, and Scaling Methods

不同策略模型、PRMs 和缩放方法的 TTS 完整结果如图 10 和图 11 所示。

A. Prompt Template for Test-Time Scaling

The system prompt for Llama 3 series models (Dubey et al., 2024) and Qwen2.5 series models (Yang et al., 2024b) are listed in Table 7 and Table 8, respectively. Following Beeching et al. (2024), we use the system prompt of the official evaluation⁶ for Llama 3 to prevent performance drop.

Table 7: System prompt for Llama 3 series models.

```
Solve the following math problem efficiently and clearly:

- For simple problems (2 steps or fewer):
Provide a concise solution with minimal explanation.

- For complex problems (3 steps or more):
Use this step-by-step format:

## Step 1: [Concise description]
[Brief explanation and calculations]

## Step 2: [Concise description]
[Brief explanation and calculations]

...

Regardless of the approach, always conclude with:

Therefore, the final answer is:  $\boxed{\text{answer}}$ . I hope it is correct.

Where [answer] is just the final number or expression that solves the problem.
```

Table 8: System prompt for Qwen2.5 series models.

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

⁶<https://huggingface.co/datasets/meta-llama/Llama-3.2-1B-Instruct-evals>

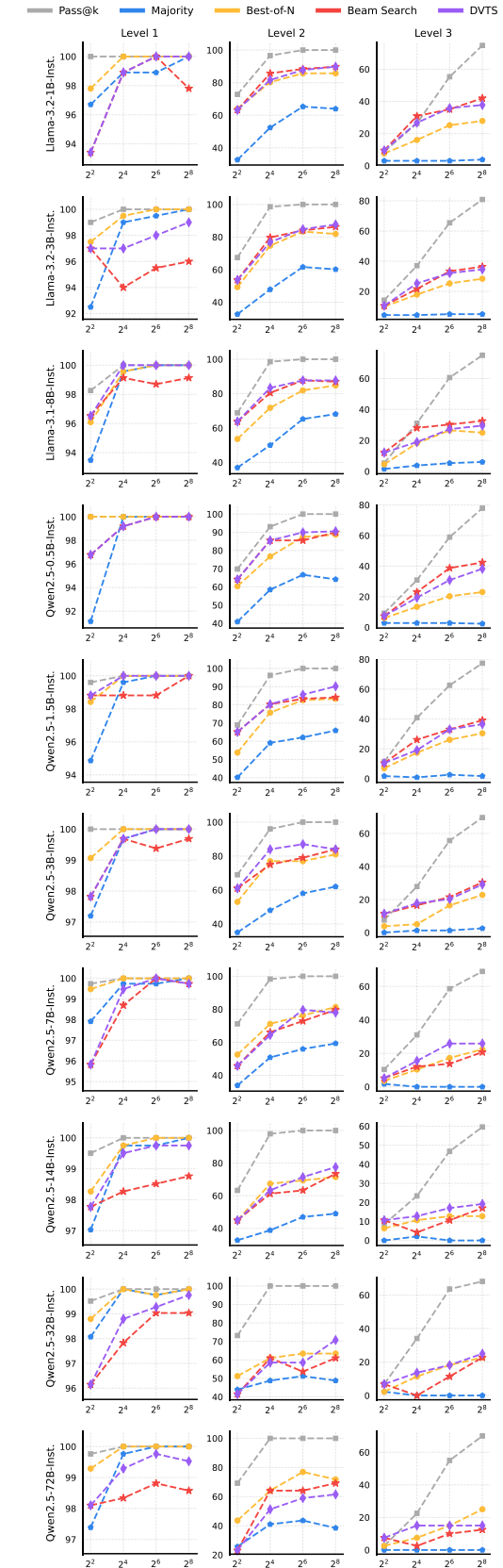


Figure 9: 三个Llama策略模型在MATH-500上不同难度级别的TTS性能。

B. Full Results of Test-Time Scaling with Different Policy Models, PRMs, and Scaling Methods

The full results of TTS with different policy models, PRMs, and scaling methods are shown in Figure 10 and Figure 11.

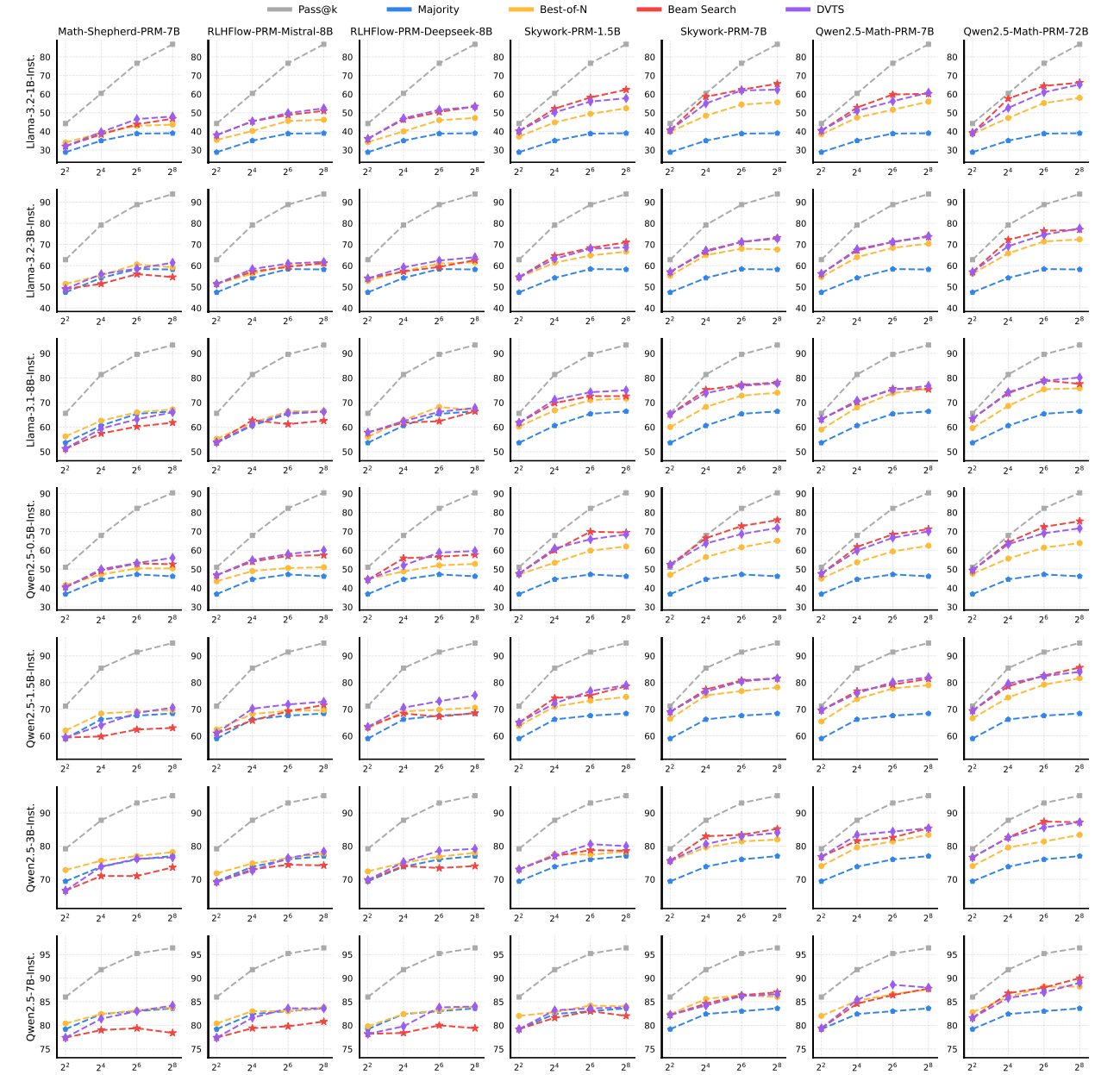


Figure 10: 不同策略模型在MATH-500上使用不同的PRMs和缩放策略的TTS性能。

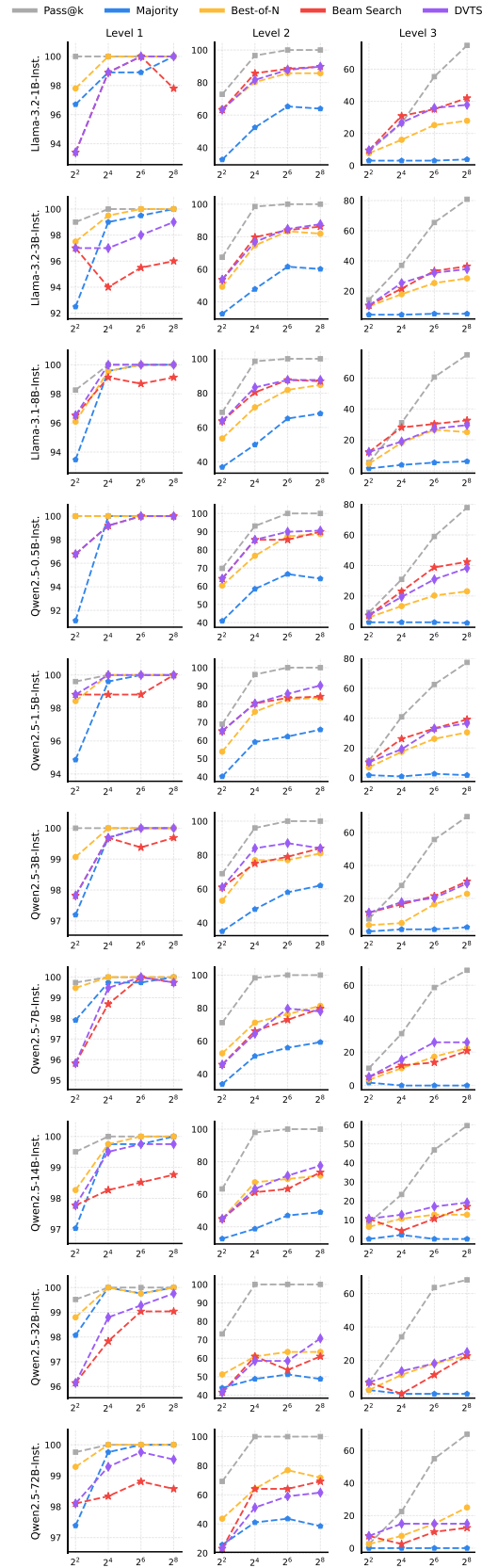


Figure 9: TTS performance of three Llama policy models on MATH-500 with different difficulty levels.

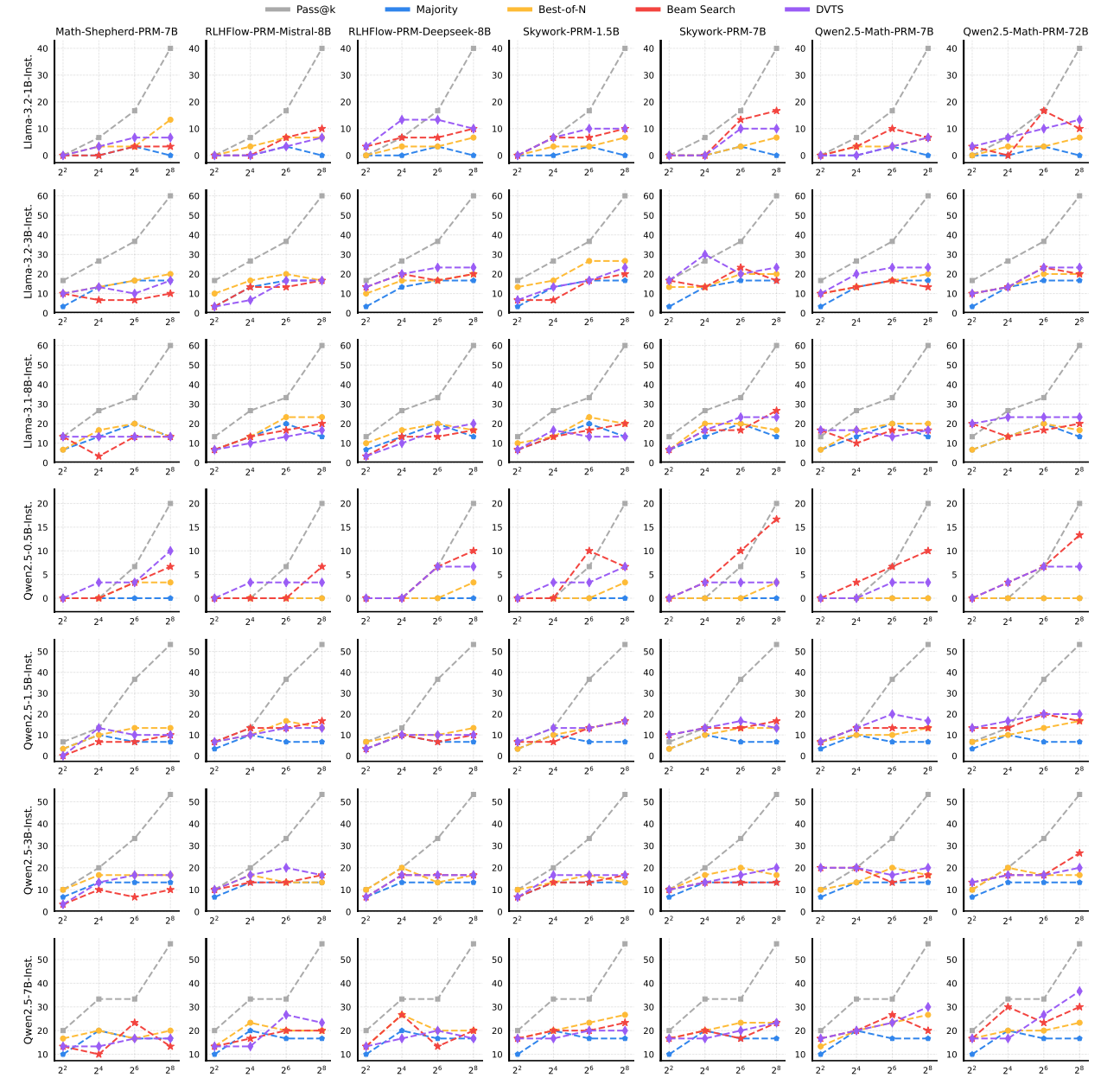


Figure 11: 不同策略模型在AIME24上使用不同的PRMs和缩放策略的TTS性能。

C. Cases

在本节中，我们为TTS提供案例并总结了PRMs的几个问题。通过分析TTS的输出，我们识别出PRMs的几个主要问题。具体来说，我们观察到四个主要类别：(1) **过度批评**：如图 13所示，PRM即使对数学上正确的步骤也给予低分，导致假阴性。(2) **错误忽略**：如图 14和图 15所示，PRM有时对明显存在数学错误的步骤给予相对较高的分数，在推理过程中未能检测到这些错误。(3) **错误定位偏差**：如图 16所示，PRM对某些中间步骤给予较低的分数，而这些步骤实际上并不是关键错误发生的地方。这表明评分信号与实际错误位置之间存在错位。(4) **评分偏差**：如图 17和图 18所示，某些训练偏差，例如对中间步骤的标记长度敏感，导致对同样正确的推理步骤评分存在较大差异。

值得注意的是，这些问题在OOD数据集（例如，未在PRM训练中使用的AIME24数据集）和同分布数据（例如，用于训练模型的MATH数据集）中都持续存在。这些问题扭曲了推理搜索过程，降低了整体性能，并减少了PRM辅助推理的可靠性。解决这些偏差对于未来模型架构和训练程序的改进是必要的，以提高PRMs的鲁棒性和可解释性。

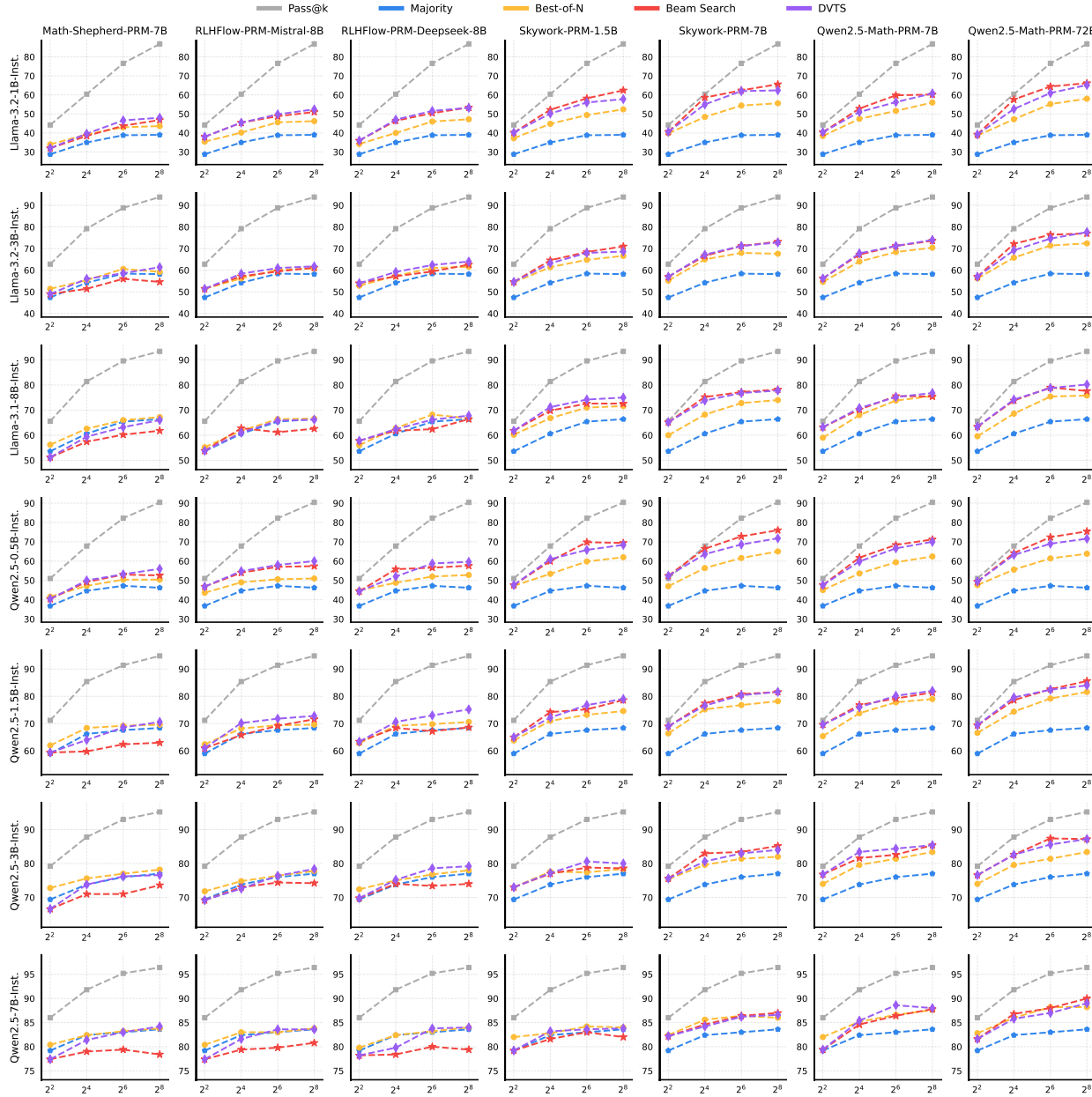


Figure 10: TTS performance of different policy models on MATH-500 with different PRMs and scaling strategies.

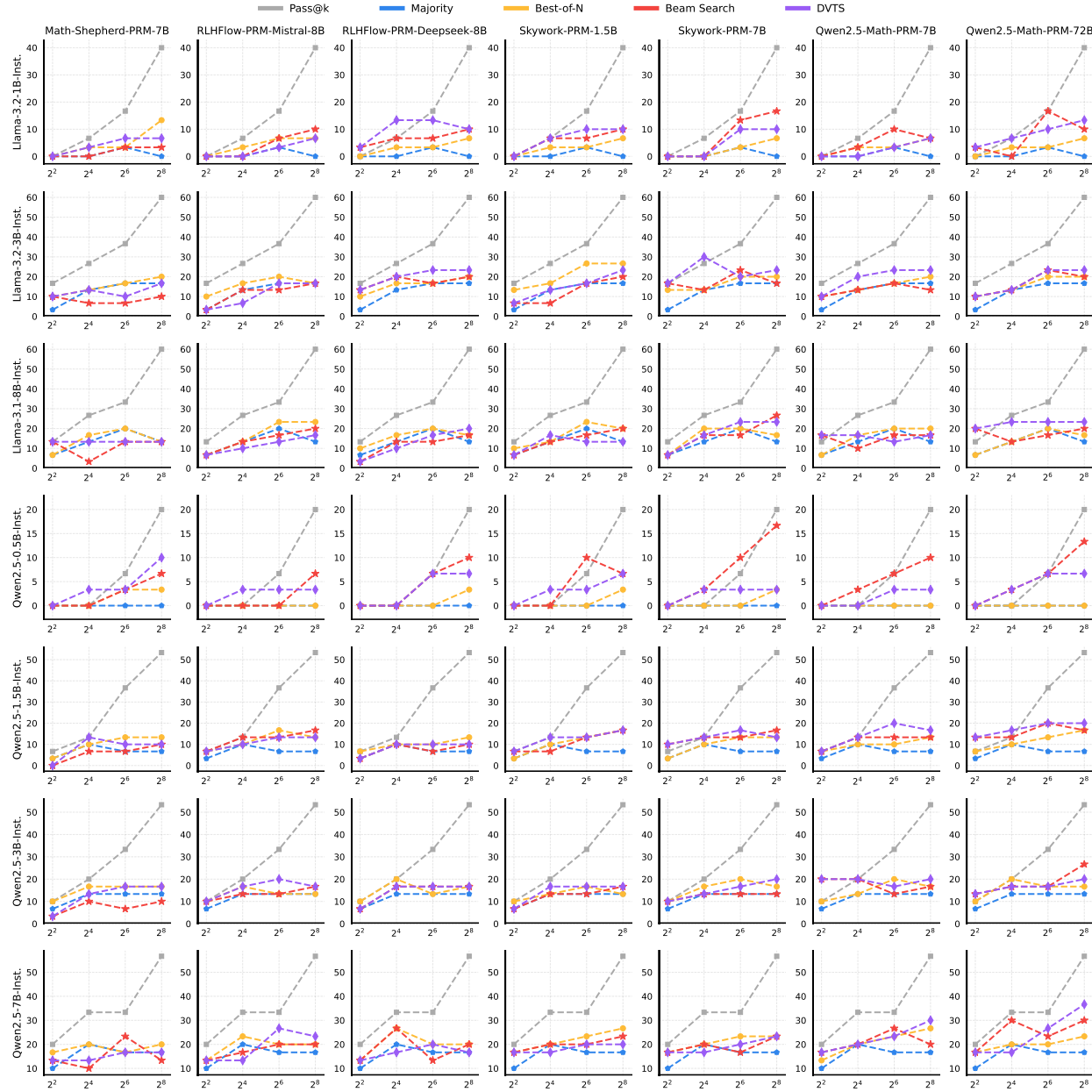


Figure 11: TTS performance of different policy models on AIME24 with different PRMs and scaling strategies.

Problem: What is the least positive integer multiple of 30 that can be written with only the digits 0 and 2?

Response (Beam Search with RLHFlow-Mistral-PRM-8B): # of Tokens: 890

Step 1: The least positive integer multiple of 30 must be divisible by 10 and 3. \nSince it can only be written with the digits 0 and 2, it must contain at least one 0 to be divisible by 10. [score=0.42]

Step 2: To be divisible by 3, the sum of its digits must also be divisible by 3.\nThe possible sums of digits are 2, 4, 6, and so on, but only 6 is divisible by 3 and achievable with the digits 2 and 0. [score=0.62]

Step 3: We can achieve a sum of 6 by using the number 220 or 022, but we cannot use leading zeros.\n\nThe number 220 is the smallest multiple of 30 that can be formed with the digits 0 and 2. \n [score=0.56]

Step 4: Multiplying 220 by 3 results in 660, which is a multiple of 30.\n\nWe've found the least positive integer multiple of 30 that can be written with only the digits 0 and 2. [score=0.87]

Therefore, the final answer is: $\boxed{660}$. [score=0.81] ❌

Response (Beam Search with RLHFlow-Deepseek-PRM-8B): # of Tokens: 2419

Step 1: Determine the divisibility requirement for the number\n\nTo be a multiple of 30, a number must be divisible by both 2 and 3. [score=0.85]

Step 2: Consider the divisibility by 2\n\nFor a number to be divisible by 2, it must end in 0, since it's the only way it will be even. [score=0.89]

Step 3: Consider the divisibility by 3\n\nFor a number to be divisible by 3, the sum of its digits must be divisible by 3. [score=0.92]

Step 4: Determine the least possible number meeting these requirements\n\nThe smallest number that can be formed with the given digits (0 and 2) and ends in 0 is 200. However, the sum of its digits ($2 + 0 + 0 = 2$) is not divisible by 3. [score=0.85]

Step 5: Find the next possible number meeting the requirements\n\nThe next number to try is 220. The sum of its digits ($2 + 2 + 0 = 4$) is still not divisible by 3. [score=0.91]

Step 6: Determine the next possible number meeting the requirements\n\nThe next number is 2200. The sum of its digits ($2 + 2 + 0 + 0 = 4$) is still not divisible by 3. [score=0.91]

Step 7: Find the next possible number meeting the requirements\n\nThe next number to try is 2020. The sum of its digits ($2 + 0 + 2 + 0 = 4$) is still not divisible by 3. [score=0.88]

Step 8: Determine the next possible number meeting the requirements\n\nThe next number is 22000. However, we can notice that the sum of the digits is still not divisible by 3. [score=0.88]

Step 9: Consider a number with more 2s\n\nInstead of adding more 0s, try adding more 2s to meet the divisibility requirement of 3. [score=0.91]

Step 10: Find the next possible number meeting the requirements\n\nThe next number to try is 2220. The sum of its digits ($2 + 2 + 2 + 0 = 6$) is divisible by 3. [score=0.85]

Therefore, the final answer is: $\boxed{2220}$. [score=0.81] ✅

Figure 12: 使用 RLHFlow-Mistral-PRM-8B 和 RLHFlow-Deepseek-PRM-8B 的束搜索玩具案例。

C. Cases

In this section, we provide cases for TTS and summarize several problems for PRMs. By analyzing the output of TTS, we identify several issues with PRMs. Specifically, we observe four major categories: (1) **Over-Criticism**: As shown in Figure 13, the PRM assigns low scores even to mathematically correct steps, resulting in false negatives. (2) **Error Neglect**: As shown in Figure 14 and Figure 15, the PRM sometimes assigns relatively high scores to steps with clear mathematical errors, failing to detect these errors during the reasoning process. (3) **Error Localization Bias**: As shown in Figure 16, the PRM assigns lower scores to certain intermediate steps that are not where the critical errors actually occur. This indicates a misalignment between the scoring signal and the actual error locations. (4) **Scoring Bias**: As shown in Figure 17 and Figure 18, certain training biases, such as sensitivity to the token length of intermediate steps, result in large discrepancies in scores for equally correct reasoning steps.

Notably, these issues persist across both OOD datasets (e.g., the AIME24 dataset, which was not used during PRM training) and in-distribution data (e.g., the MATH dataset used to train the model). These problems distort the reasoning search process, degrade overall performance, and reduce the reliability of PRM-assisted reasoning. Addressing these biases in future model architectures and training procedures is necessary to improve the robustness and interpretability of PRMs.

Problem:

Simplify $\sqrt{242}$.

Response:

To simplify $\sqrt{242}$, we can use the prime factorization method. Here are the steps: [score=0.90] ✓

1. **Prime Factorization**: First, we need to find the prime factors of 242. 242 is even, so it is divisible by 2. $242 \div 2 = 121$. So, $242 = 2 \times 121$. Next, we factor 121. We know that 121 is 11×11 . So, $121 = 11 \times 11$. So, $242 = 2 \times 11 \times 11$. [score=0.88] ✓

2. **Rewrite the Square Root**: Now, we can rewrite the square root of 242 using its prime factorization. $\sqrt{242} = \sqrt{2 \times 11 \times 11}$. [score=0.84] ✓

3. **Simplify the Square Root**: We can use the property of square roots that $\sqrt{a \times b} = \sqrt{a} \times \sqrt{b}$. $\sqrt{242} = \sqrt{2 \times 11 \times 11} = \sqrt{2} \times \sqrt{11 \times 11} = \sqrt{2} \times 11 = 11\sqrt{2}$. [score=0.53] ✓

Therefore, the simplified form of $\sqrt{242}$ is $\boxed{11\sqrt{2}}$. [score=0.46] ✓

Figure 13: TTS case of Over-Criticism.

Problem: What is the least positive integer multiple of 30 that can be written with only the digits 0 and 2?

Response (Beam Search with RLHFlow-Mistral-PRM-8B): # of Tokens: 890

Step 1: The least positive integer multiple of 30 must be divisible by 10 and 3. \nSince it can only be written with the digits 0 and 2, it must contain at least one 0 to be divisible by 10. [score=0.42]

Step 2: To be divisible by 3, the sum of its digits must also be divisible by 3.\nThe possible sums of digits are 2, 4, 6, and so on, but only 6 is divisible by 3 and achievable with the digits 2 and 0. [score=0.62]

Step 3: We can achieve a sum of 6 by using the number 220 or 022, but we cannot use leading zeros.\nThe number 220 is the smallest multiple of 30 that can be formed with the digits 0 and 2. \n [score=0.56]

Step 4: Multiplying 220 by 3 results in 660, which is a multiple of 30.\nWe've found the least positive integer multiple of 30 that can be written with only the digits 0 and 2. [score=0.87]

Therefore, the final answer is: $\boxed{660}$. [score=0.81] ✗

Response (Beam Search with RLHFlow-Deepseek-PRM-8B): # of Tokens: 2419

Step 1: Determine the divisibility requirement for the number\nTo be a multiple of 30, a number must be divisible by both 2 and 3. [score=0.85]

Step 2: Consider the divisibility by 2\nFor a number to be divisible by 2, it must end in 0, since it's the only way it will be even. [score=0.89]

Step 3: Consider the divisibility by 3\nFor a number to be divisible by 3, the sum of its digits must be divisible by 3. [score=0.92]

Step 4: Determine the least possible number meeting these requirements\nThe smallest number that can be formed with the given digits (0 and 2) and ends in 0 is 200. However, the sum of its digits (2 + 0 + 0 = 2) is not divisible by 3. [score=0.85]

Step 5: Find the next possible number meeting the requirements\nThe next number to try is 220. The sum of its digits (2 + 2 + 0 = 4) is still not divisible by 3. [score=0.91]

Step 6: Determine the next possible number meeting the requirements\nThe next number is 2200. The sum of its digits (2 + 2 + 0 + 0 = 4) is still not divisible by 3. [score=0.91]

Step 7: Find the next possible number meeting the requirements\nThe next number to try is 2020. The sum of its digits (2 + 0 + 2 + 0 = 4) is still not divisible by 3. [score=0.88]

Step 8: Determine the next possible number meeting the requirements\nThe next number is 22000. However, we can notice that the sum of the digits is still not divisible by 3. [score=0.88]

Step 9: Consider a number with more 2s\nInstead of adding more 0s, try adding more 2s to meet the divisibility requirement of 3. [score=0.91]

Step 10: Find the next possible number meeting the requirements\nThe next number to try is 2220. The sum of its digits (2 + 2 + 2 + 0 = 6) is divisible by 3. [score=0.85]

Therefore, the final answer is: $\boxed{2220}$. [score=0.81] ✓

Problem:

Suppose $\sin D = 0.7$ in the diagram below. What is DE ? [asy]\npair D,E,F;\nF = (0,0);\nD = (sqrt(51),7);\nE = (0,7);\ndraw(D--E--F);\ndraw(rightanglemark(D,E,F,15));\nlabel("\$D\$",D,NE);\nlabel("\$E\$",E,NW);\nlabel("\$F\$",F,SW);\nlabel("\$7\$", (E+F)/2,W);\n[/asy]

Response:

To solve for DE given that $\sin D = 0.7$ and the length $EF = 7$, we can follow these steps: [score=0.99] ✓

1. **Identify the given information and the right triangle:** We are given $\sin D = 0.7$. $\angle DEF$ is a right angle. $EF = 7$. [score=0.99] ✓
 2. **Understand the trigonometric relationship:** In a right triangle, \sin of an angle is the ratio of the length of the opposite side to the hypotenuse. Here, $\sin D = \frac{EF}{DE}$. [score=0.90] ✗
 3. **Set up the equation using the given $\sin D$:** $\sin D = \frac{EF}{DE}$ Substituting the given values: $0.7 = \frac{7}{DE}$ [score=0.97] ✗
 4. **Solve for DE :** To isolate DE , multiply both sides of the equation by DE : $0.7 \cdot DE = 7$ Divide both sides by 0.7: $DE = \frac{7}{0.7}$ Simplify the division: $DE = 10$ [score=0.99] ✗
- Therefore, the length of DE is $\boxed{10}$. [score=0.99] ✗

Figure 14: TTS case of Error Neglect.

Figure 12: Toy case of beam search with RLHFlow-Mistral-PRM-8B and RLHFlow-Deepseek-PRM-8B.

Problem:Simplify $\sqrt{242}$.**Response:**

To simplify $\sqrt{242}$, we can use the prime factorization method. Here are the steps: [score=0.90] ✓

1. **Prime Factorization**: First, we need to find the prime factors of 242. 242 is even, so it is divisible by 2. $242 \div 2 = 121$. So, $242 = 2 \times 121$. Next, we factor 121. We know that 121 is 11×11 . So, $242 = 2 \times 11 \times 11$. [score=0.88] ✓

2. **Rewrite the Square Root**: Now, we can rewrite the square root of 242 using its prime factorization. $\sqrt{242} = \sqrt{2 \times 11 \times 11}$. [score=0.84] ✓

3. **Simplify the Square Root**: We can use the property of square roots that $\sqrt{a \times b} = \sqrt{a} \times \sqrt{b}$. $\sqrt{242} = \sqrt{2 \times 11 \times 11} = \sqrt{2} \times \sqrt{11 \times 11} = \sqrt{2} \times 11 = 11\sqrt{2}$. [score=0.53] ✓

Therefore, the simplified form of $\sqrt{242}$ is $11\sqrt{2}$. [score=0.46] ✓

Figure 13: TTS case of Over-Criticism.

Problem:

Suppose $\sin D = 0.7$ in the diagram below. What is DE ? [asy] \npair D,E,F; \nF = (0,0); \nD = (sqrt(51),7); \nE = (0,7); \ndraw(D--E--F--D); \ndraw(rightanglemark(D,E,F,15)); \n label("\$D\$",D,NE); \n label("\$E\$",E,NW); \n label("\$F\$",F,SW); \n label("\$7\$", (E+F)/2, W); \n [/asy]

Response:

To find the length of DE , we start by analyzing the given right triangle DEF where $\angle EDF = 90^\circ$. We know that the length of EF is 7 and the sine of angle D is 0.7. We can use the definition of sine in a right triangle, which is the ratio of the length of the opposite side to the hypotenuse. [score=0.74] ✓

Given: $\sin D = \frac{EF}{DE} = 0.7$. Since $EF = 7$, we substitute to get: $\frac{7}{DE} = 0.7$. To find DE , we solve the equation: $DE = \frac{7}{0.7} = \frac{7}{\frac{7}{10}} = 10$. [score=0.88] ✗

Thus, the length of DE is: $\boxed{10}$. [score=0.82] ✗

Figure 15: TTS case of Error Neglect.

Problem:

Suppose $\sin D = 0.7$ in the diagram below. What is DE ? [asy]\npair D,E,F;\nF = (0,0);\nD = (sqrt(51),7);\nE = (0,7);\ndraw(D--E--F--D);\ndraw(rightanglemark(D,E,F,15));\nlabel("\$D\$",D,NE);\nlabel("\$E\$",E,NW);\nlabel("\$F\$",F,SW);\nlabel("\$7\$", (E+F)/2,W);\n[/asy]

Response:

To solve for DE given that $\sin D = 0.7$ and the length $EF = 7$, we can follow these steps: [score=0.99] ✓

1. **Identify the given information and the right triangle:** We are given $\sin D = 0.7$. $\angle DEF$ is a right angle. $EF = 7$. [score=0.99] ✓
2. **Understand the trigonometric relationship:** In a right triangle, \sin of an angle is the ratio of the length of the opposite side to the hypotenuse. Here, $\sin D = \frac{EF}{DE}$. [score=0.90] ✗
3. **Set up the equation using the given $\sin D$:** $\sin D = \frac{EF}{DE}$. Substituting the given values: $0.7 = \frac{7}{DE}$. [score=0.97] ✗
4. **Solve for DE :** To isolate DE , multiply both sides of the equation by DE : $0.7 \cdot DE = 7$. Divide both sides by 0.7: $DE = \frac{7}{0.7}$. Simplify the division: $DE = 10$. [score=0.99] ✗

Therefore, the length of DE is $\boxed{10}$. [score=0.99] ✗

Figure 14: TTS case of Error Neglect.

Problem:

Let ABC be a triangle inscribed in circle ω . Let the tangents to ω at B and C intersect at point D , and let \overline{AD} intersect ω at P . If $AB=5$, $BC=9$, and $AC=10$, AP can be written as the form $\frac{m}{n}$, where m and n are relatively prime integers. Find $m + n$.

Response:

To solve the problem, we start by noting that D is the intersection of the tangents to the circumcircle ω at B and C . This means that D is the excenter of the triangle ABC opposite to A . By the properties of tangents from a point to a circle, $DB = DC$. [score=0.20] ✓

We use the fact that AD is the angle bisector of $\angle BAC$ when extended to intersect the circumcircle again at P . By the Angle Bisector Theorem, we know that the ratio of the segments created by the angle bisector on BC is equal to the ratio of the other two sides of the triangle. Therefore, we have: $\frac{BD}{DC} = \frac{AB}{AC} = \frac{5}{10} = \frac{1}{2}$. Since $DB = DC$, we can use the Power of a Point theorem at D . The power of point D with respect to the circle is: $DB^2 = DC^2 = DA \cdot DP$. Since D is the intersection of the tangents at B and C , we have: $DB^2 = DC^2 = BD^2 = BD \cdot BP$. Let $BD = x$. Then $DC = x$ and $BC = BD + DC = 2x = 9$, so $x = \frac{9}{2}$. Therefore, $BD = DC = \frac{9}{2}$. [score=0.66] ✗

Using the Power of a Point theorem again, we have: $BD^2 = DA \cdot DP$ implies $\left(\frac{9}{2}\right)^2 = DA \cdot DP$ implies $\frac{81}{4} = DA \cdot DP$. We need to find the length of DA . By the Angle Bisector Theorem, the ratio of the segments BD and DC is the same as the ratio of AB and AC , which is $\frac{1}{2}$. Therefore, DA can be found using the formula for the length of the angle bisector: $AD = \frac{2 \cdot AB \cdot AC \cdot \cos \left(\frac{A}{2}\right)}{AB + AC}$. However, a simpler approach is to use the fact that $AD \cdot DP = \left(\frac{9}{2}\right)^2 = \frac{81}{4}$. Since $AD + DP = AP$, we need to find AP . Using the fact that $DA \cdot DP = \frac{81}{4}$ and knowing that $DA = \frac{90}{13}$ (from the Angle Bisector Theorem and the length of the angle bisector formula), we can solve for AP : $AP = \frac{81}{4} \cdot \frac{13}{90} = \frac{81 \cdot 13}{4 \cdot 90} = \frac{1053}{360} = \frac{117}{40}$. Thus, $AP = \frac{117}{40}$, and the sum $m + n$ is: $117 + 40 = 157$. Therefore, the answer is: $\boxed{157}$. [score=0.92] ✗

Figure 16: TTS case of Error Localization Bias.

Problem:

Suppose $\sin D = 0.7$ in the diagram below. What is DE ? [asy]\npair D,E,F;\nF = (0,0);\nD = (sqrt(51),7);\nE = (0,7);\ndraw(D--E--F--D);\ndraw(rightanglemark(D,E,F,15));\nlabel("\$D\$",D,NE);\nlabel("\$E\$",E,NW);\nlabel("\$F\$",F,SW);\nlabel("\$7\$", (E+F)/2, W);\n[/asy]

Response:

To find the length of DE , we start by analyzing the given right triangle DEF where $\angle EDF = 90^\circ$. We know that the length of EF is 7 and the sine of angle D is 0.7. We can use the definition of sine in a right triangle, which is the ratio of the length of the opposite side to the hypotenuse. [score=0.74] ✓

Given: $\sin D = \frac{EF}{DE} = 0.7$ Since $EF = 7$, we substitute to get: $\frac{7}{DE} = 0.7$ To find DE , we solve the equation: $DE = \frac{7}{0.7} = \frac{7}{\frac{7}{10}} = 10$ [score=0.88] ✗

Thus, the length of DE is: $\boxed{10}$ [score=0.82] ✗

Figure 15: TTS case of Error Neglect.

Problem:

You have seven bags of gold coins. Each bag has the same number of gold coins. One day, you find a bag of 53 coins. You decide to redistribute the number of coins you have so that all eight bags you hold have the same number of coins. You successfully manage to redistribute all the coins, and you also note that you have more than 200 coins. What is the smallest number of coins you could have had before finding the bag of 53 coins?

Response:

Let's denote the number of gold coins in each of the seven bags initially as n . Therefore, the total number of coins before finding the bag of 53 coins is $7n$. After finding the bag of 53 coins, the total number of coins becomes $7n + 53$.

[score=0.98] ✓

Since you redistribute the coins so that all eight bags have the same number of coins, the total number of coins must be divisible by 8. Therefore, $7n + 53$ must be divisible by 8. We can express this condition as: $7n + 53 \equiv 0 \pmod{8}$. Since $7n \equiv -53 \pmod{8}$, we have: $7n \equiv 3 \pmod{8}$. [score=0.81] ✓

To solve for n , we need the multiplicative inverse of 7 modulo 8. The multiplicative inverse of 7 modulo 8 is a number x such that: $7x \equiv 1 \pmod{8}$. By testing values, we find that $x = 7$ works because: $7 \cdot 7 = 49 \equiv 1 \pmod{8}$. Thus, multiplying both sides of $7n \equiv 3 \pmod{8}$ by 7, we get: $n \equiv 21 \pmod{8}$. So, $n \equiv 5 \pmod{8}$. Therefore, $n = 8k + 5$ for some integer k . [score=0.92] ✓

The total number of coins after finding the bag of 53 coins is $7n + 53 = 7(8k + 5) + 53 = 56k + 35 + 53 = 56k + 88$. Since the number of coins must be more than 200, we have: $56k + 88 > 200$. Solving for k , we get: $56k > 112$, so $k > 2$. The smallest integer k that satisfies this inequality is $k = 3$. Substituting $k = 3$ into the expression for the total number of coins, we get: $56 \cdot 3 + 88 = 168 + 88 = 256$. [score=0.79] ✓

To find the number of coins before finding the bag of 53 coins, we calculate: $7n = 7 \cdot (8 \cdot 3 + 5) = 7 \cdot 29 = 203$. [score=0.51] ✓

Thus, the smallest number of coins you could have had before finding the bag of 53 coins is $\boxed{203}$. [score=0.68] [token num=31] ✓

Figure 17: TTS case of Scoring Bias.

Problem:

Let $\triangle ABC$ be a triangle inscribed in circle ω . Let the tangents to ω at B and C intersect at point D , and let \overline{AD} intersect ω at P . If $AB=5$, $BC=9$, and $AC=10$, AP can be written as the form $\frac{m}{n}$, where m and n are relatively prime integers. Find $m + n$.

Response:

To solve the problem, we start by noting that D is the intersection of the tangents to the circumcircle ω at B and C . This means that D is the excenter of the triangle ABC opposite to A . By the properties of tangents from a point to a circle, $DB = DC$. [score=0.20] ✓

We use the fact that AD is the angle bisector of $\angle BAC$ when extended to intersect the circumcircle again at P . By the Angle Bisector Theorem, we know that the ratio of the segments created by the angle bisector on BC is equal to the ratio of the other two sides of the triangle. Therefore, we have: $\frac{BD}{DC} = \frac{AB}{AC} = \frac{5}{10} = \frac{1}{2}$. Since $DB = DC$, we can use the Power of a Point theorem at D . The power of point D with respect to the circle is: $DB^2 = DC^2 = DA \cdot DP$. Since D is the intersection of the tangents at B and C , we have: $DB^2 = DC^2 = BD \cdot DC$. Let $BD = x$. Then $DC = 2x$ and $BC = BD + DC = 3x = 9$, so $x = 3$. Therefore, $BD = DC = 3$. [score=0.66] ✗

Using the Power of a Point theorem again, we have: $BD^2 = DA \cdot DP$ implies $\left(\frac{9}{2}\right)^2 = DA \cdot DP$ implies $\frac{81}{4} = DA \cdot DP$. We need to find the length of DA . By the Angle Bisector Theorem, the ratio of the segments BD and DC is the same as the ratio of AB and AC , which is $\frac{1}{2}$. Therefore, DA can be found using the formula for the length of the angle bisector: $AD = \frac{2 \cdot AB \cdot AC \cdot \cos\left(\frac{A}{2}\right)}{AB + AC}$. However, a simpler approach is to use the fact that $AD \cdot DP = \left(\frac{9}{2}\right)^2 = \frac{81}{4}$. Since $AD + DP = AP$, we need to find AP . Using the fact that $DA \cdot DP = \frac{81}{4}$ and knowing that $DA = \frac{90}{13}$ (from the Angle Bisector Theorem and the length of the angle bisector formula), we can solve for AP : $\frac{90}{13} \cdot DP = \frac{81}{4}$ implies $DP = \frac{81 \cdot 13}{4 \cdot 90} = \frac{1053}{360} = \frac{117}{40}$. Thus, $AP = \frac{117}{40}$, and the sum $m + n$ is: $117 + 40 = 157$. Therefore, the answer is: $\boxed{157}$. [score=0.92] ✗

Figure 16: TTS case of Error Localization Bias.

Problem:

You have seven bags of gold coins. Each bag has the same number of gold coins. One day, you find a bag of 53 coins. You decide to redistribute the number of coins you have so that all eight bags you hold have the same number of coins. You successfully manage to redistribute all the coins, and you also note that you have more than 200 coins. What is the smallest number of

Response:

Let's denote the number of gold coins in each of the seven bags initially as x . Therefore, the total number of coins initially is $7x$. After finding a bag of 53 coins, the total number of coins becomes $7x + 53$. [score=0.98] ✓

We are given that the total number of coins can be evenly distributed among 8 bags, so $7x + 53$ must be divisible by 8. Additionally, we know that the total number of coins is more than 200. We need to find the smallest possible value of $7x + 53$ that satisfies both conditions. [score=0.95] ✓

First, we express the condition that $7x + 53$ is divisible by 8: $7x + 53 \equiv 0 \pmod{8}$. Since $7 \equiv -1 \pmod{8}$, we can rewrite the equation as: $-x + 53 \equiv 0 \pmod{8}$ or $-x \equiv -53 \pmod{8}$ or $x \equiv 53 \pmod{8}$. This means that x can be written in the form: $x = 8k + 5$ for some integer k . [score=0.90] ✓

Next, we substitute $x = 8k + 5$ into the expression for the total number of coins: $7x + 53 = 7(8k + 5) + 53 = 56k + 35 + 53 = 56k + 88$. We need $56k + 88$ to be more than 200: $56k + 88 > 200$ or $56k > 112$ or $k > 2$. The smallest integer k that satisfies this inequality is $k = 3$.

Substituting $k = 3$ into the expression for the total number of coins, we get: $56 \cdot 3 + 88 = 168 + 88 = 256$. Thus, the smallest number of coins you could have had before finding the bag of 53 coins is: $7x = 7(8 \cdot 3 + 5) = 7 \cdot 29 = 203$. Therefore, the smallest number of coins you could have had before finding the bag of 53 coins is $\boxed{203}$. [score=0.12] [token num=283] ✓

Figure 18: TTS case of Scoring Bias.

Problem:

You have seven bags of gold coins. Each bag has the same number of gold coins. One day, you find a bag of 53 coins. You decide to redistribute the number of coins you have so that all eight bags you hold have the same number of coins. You successfully manage to redistribute all the coins, and you also note that you have more than 200 coins. What is the smallest number of coins you could have had before finding the bag of 53 coins?

Response:

Let's denote the number of gold coins in each of the seven bags initially as n . Therefore, the total number of coins before finding the bag of 53 coins is $7n$. After finding the bag of 53 coins, the total number of coins becomes $7n + 53$.

[score=0.98] ✓

Since you redistribute the coins so that all eight bags have the same number of coins, the total number of coins must be divisible by 8. Therefore, $7n + 53$ must be divisible by 8. We can express this condition as: $7n + 53 \equiv 0 \pmod{8}$. Since $7n \equiv -53 \pmod{8}$, we have: $7n \equiv 3 \pmod{8}$. [score=0.81] ✓

To solve for n , we need the multiplicative inverse of 7 modulo 8. The multiplicative inverse of 7 modulo 8 is a number x such that: $7x \equiv 1 \pmod{8}$. By testing values, we find that $x = 7$ works because: $7 \cdot 7 = 49 \equiv 1 \pmod{8}$. Thus, multiplying both sides of $7n \equiv 3 \pmod{8}$ by 7, we get: $n \equiv 21 \pmod{8}$. So, $n \equiv 5 \pmod{8}$. [score=0.92] ✓

The total number of coins after finding the bag of 53 coins is $7n + 53 = 7(8k + 5) + 53 = 56k + 35 + 53 = 56k + 88$. Since the number of coins must be more than 200, we have: $56k + 88 > 200$. The smallest integer k that satisfies this inequality is $k = 3$. Substituting $k = 3$ into the expression for the total number of coins, we get: $56 \cdot 3 + 88 = 168 + 88 = 256$. [score=0.79] ✓

To find the number of coins before finding the bag of 53 coins, we calculate: $7n = 7 \cdot (8 \cdot 3 + 5) = 7 \cdot 29 = 203$. [score=0.51] ✓

Thus, the smallest number of coins you could have had before finding the bag of 53 coins is $\boxed{203}$. [score=0.68] [token num=31] ✓

Figure 17: TTS case of Scoring Bias.

Problem:

You have seven bags of gold coins. Each bag has the same number of gold coins. One day, you find a bag of 53 coins. You decide to redistribute the number of coins you have so that all eight bags you hold have the same number of coins. You successfully manage to redistribute all the coins, and you also note that you have more than 200 coins. What is the smallest number of

Response:

Let's denote the number of gold coins in each of the seven bags initially as x .

Therefore, the total number of coins initially is $7x$. After finding a bag of 53 coins, the total number of coins becomes $7x + 53$. [score=0.98] ✓

We are given that the total number of coins can be evenly distributed among 8 bags, so $7x + 53$ must be divisible by 8. Additionally, we know that the total number of coins is more than 200. We need to find the smallest possible value of $7x + 53$ that satisfies both conditions. [score=0.95] ✓

First, we express the condition that $7x + 53$ is divisible by 8:
 $7x + 53 \equiv 0 \pmod{8}$
 Since $7 \equiv -1 \pmod{8}$, we can rewrite the equation as:
 $-x + 53 \equiv 0 \pmod{8}$
 $-x \equiv -53 \pmod{8}$
 $x \equiv 53 \pmod{8}$
 $x \equiv 5 \pmod{8}$
 This means that x can be written in the form: $x = 8k + 5$ for some integer k . [score=0.90] ✓

Next, we substitute $x = 8k + 5$ into the expression for the total number of coins:
 $7x + 53 = 7(8k + 5) + 53 = 56k + 35 + 53 = 56k + 88$
 We need $56k + 88$ to be more than 200:
 $56k + 88 > 200$
 $56k > 112$
 $k > 2$
 The smallest integer k that satisfies this inequality is $k = 3$.

Substituting $k = 3$ into the expression for the total number of coins, we get:
 $56 \cdot 3 + 88 = 168 + 88 = 256$
 Thus, the smallest number of coins you could have had before finding the bag of 53 coins is:
 $7x = 7(8 \cdot 3 + 5) = 7 \cdot 29 = 203$
 Therefore, the smallest number of coins you could have had before finding the bag of 53 coins is $\boxed{203}$. [score=0.12] [token num=283] ✓

Figure 18: TTS case of Scoring Bias.