An Empirical Analysis of Compute-Optimal Inference for Problem-Solving with Language Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

The optimal training configurations of large language models (LLMs) with respect to model sizes and compute budgets have been extensively studied. But how to optimally configure LLMs during inference has not been explored in sufficient depth. We study *compute-optimal inference*: designing models and inference strategies that optimally trade off additional inference-time compute for improved performance. As a first step towards understanding and designing compute-optimal inference methods, we assessed the effectiveness and computational efficiency of multiple inference strategies such as Greedy Search, Majority Voting, Best-of-N, Weighted Voting, and their variants on two different Tree Search algorithms, involving different model sizes (e.g., 7B and 34B) and computational budgets. We found that a smaller language model with a novel tree search algorithm typically achieves a Pareto-optimal trade-off. These results highlight the potential benefits of deploying smaller models equipped with more sophisticated decoding algorithms in end-devices to enhance problem-solving accuracy. For instance, we show that the Llemma-7B model can achieve competitive accuracy to a Llemma-34B model on MATH500 while using $2\times$ less FLOPs. Our findings could potentially apply to any generation task with a well-defined measure of success.

1 Introduction

Scaling laws of neural networks [Hestness et al., 2017, Rosenfeld et al., 2019] have been established across a range of domains, including language modeling [Kaplan et al., 2020, Hoffmann et al., 2022, OpenAI, 2023], image modeling [Henighan et al., 2020, Yu et al., 2022, Peebles and Xie, 2023], video modeling [Brooks et al., 2024], reward modeling [Gao et al., 2023], and board games [Jones, 2021]. These studies have demonstrated how model performance is influenced by both the size of the model and the amount of training computation. However, there is limited knowledge on how varying the compute during inference affects model performance after the model has been trained.

To improve the task performance of large language models (LLMs), inference techniques typically involve additional computation in a *performance maximization* step at inference time [Nye et al., 2021, Wei et al., 2022, Wang et al., 2022b, Yao et al., 2023, Chen et al., 2024b]. This cost must be taken into account for *compute-optimal inference.* For example, a Monte Carlo Tree Search (MCTS) method [Jones, 2021] may improve task performance, but potentially cost much more than simply sampling solutions multiple times. Generally speaking, we need a comprehensive understanding of how various inference-time methods (e.g., Best-of-N, majority voting) trade off between performance and cost. To improve our understanding, this paper presents a thorough empirical evaluation with careful analysis over various configurations of representative LLMs and inference algorithms.

Specifically, we explore how to select an optimal model size (e.g., 7B or 34B) for the policy model and an effective inference strategy (e.g., Greedy Search, Majority Voting, Best-of-N, Weighted Voting,

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

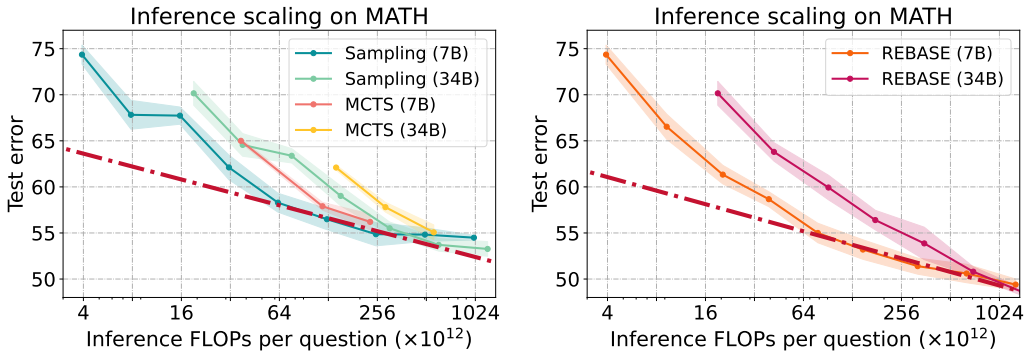

Figure 1: **The inference computation scaling laws** exhibited in error rate on the **MATH500** test set based on weighted majority voting, where the left figure shows sampling vs. MCTS, and the right figure shows our proposed REBASE. Clearly, the error rate decreases steadily when the computation increases, and REBASE exhibits a Pareto-optimal tradeoff during inference.

and their Tree Search variants) to maximize performance (i.e., accuracy) within a given compute budget. We manipulate the inference computation (FLOPs) of a fixed model by generating additional tokens through the policy model, sampling further candidate solutions, and ranking them with a reward model. We analyze the performance of a family of math-specialized LLMs (i.e., Llemma-7B and Llemma-34B [Azerbayev et al., 2023]) fine-tuned on the MetaMath dataset [Yu et al., 2023] and measure the error rate on the GSM8K test set [Cobbe et al., 2021a] and MATH500 test set [Hendrycks et al., 2021b, Lightman et al., 2023b].

Our analysis shows that voting-based methods generally outperform the strategy which selects the best solution (i.e., Best-of-N), and weighted voting has the most favorable results (Section 4.3, Figure 5 & 6). However, neither method shows a desirable behavior at high levels of compute. For instance, weighted voting saturates when sampling more than 128 solutions (Figure 1). We have also found that the commonly used MCTS method does not perform well with weighted voting, as it often yields many unfinished solutions, hence having less votes. To address this issue, we propose a novel tree search algorithm, *REward BAlanced SEarch (*REBASE*)*, which pairs well with weighted voting and improves the Pareto-optimal trade-off between accuracy and inference compute. The key idea of REBASE is to use a node-quality based reward to control the exploitation and pruning properties of tree search, while ensuring enough candidate solutions for voting or selection.

In our experiments, REBASE consistently outperforms sampling and MCTS methods across all settings, models, and tasks. Importantly, we find that REBASE with a *smaller* language model typically achieves a Pareto-optimal trade-off. For instance, we show that the Llemma-7B model can achieve competitive accuracy to a Llemma-34B model while using 2× less FLOPs when evaluating on MATH500 (Figure 1) or GSM8K (Figure 4). These findings underscore the advantages of using smaller models with advanced inference-time algorithms on end-devices to improve problem-solving.

2 Related Works

Mathematical Reasoning with LLMs. Large language models have made significant progress in recent years, and have exhibited strong reasoning abilities [Brown et al., 2020, Hoffmann et al., 2022, Chowdhery et al., 2022, Lewkowycz et al., 2022]. Mathematical problem solving is a key task for measuring LLM reasoning abilities [Cobbe et al., 2021a, Hendrycks et al., 2021b]. [Ling et al., 2017] first developed the method of producing step by step solutions that lead to the final answer. Later, [Cobbe et al., 2021b] extended the work by finetuning the pre-trained language model on a large dataset to solve math word problems, a verifier is trained for evaluating solutions and ranking solutions. Nye et al. [2021] train models to use a scratchpad and improve their performance on algorithmic tasks. Wei et al. [2022] demonstrate that the reasoning ability of a language model can be elicited through the prompting. Subsequent research [Kojima et al., 2022, Lewkowycz et al., 2022, Zhou et al., 2022] in reasoning tasks has also highlighted the efficacy of rationale augmentation. We choose problem solving in mathematics as the task to study the compute-optimal strategy since it allows us to accurately evaluate the problem solving ability of LLMs.

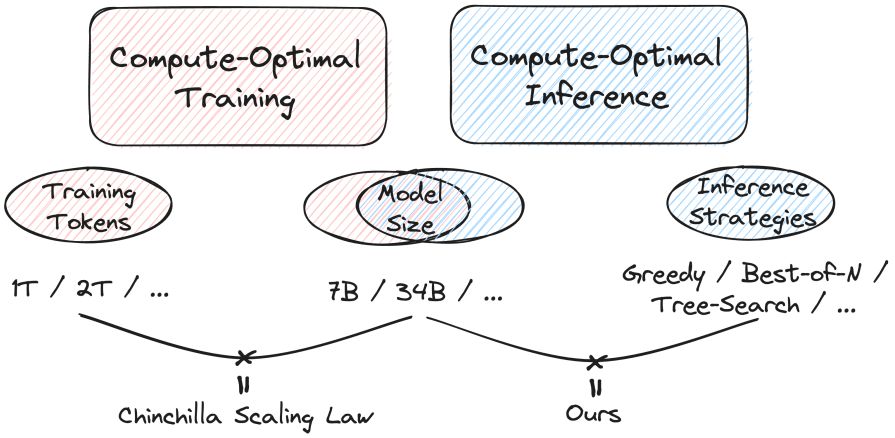

Figure 2: **Illustration of compute-optimal scaling laws in training and inference.** The Chinchilla scaling law shows how to choose a model size and number of training tokens under a training-compute budget, while ours shows how to choose a model size and an inference strategy under a inference-compute budget.

Inference Strategies of LLM Problem Solving. A variety of inference (also called decoding) strategies have been developed to generate sequences with a trained model. Deterministic methods such as greedy decoding and beam search [Teller, 2000, Graves, 2012] find highly probable sequences, often yielding high quality results but without diversity. Sampling algorithms (e.g., temperature sampling [Ackley et al., 1985]) can produce a diverse set of results which are then aggregated to achieve higher accuracy (e.g., via majority voting [Wang et al., 2022a]). Recent methods combine search algorithms with modern LLMs, including breadth-first or depth-first search [Yao et al., 2023], Monte-Carlo Tree Search (MCTS) [Zhang et al., 2023, Zhou et al., 2023, Liu et al., 2024, Choi et al., 2023], and Self-evaluation Guided Beam Search [Xie et al., 2023]. All of these methods show that using search at inference time can lead to performance gains in various tasks. However, the trade-off for the improved performance is the use of compute to perform the search. Analyzing the trade-off between compute budget and LLM inference performance remains understudied. In this paper, we systematically analyze the trade-off between compute budget and problem-solving performance, and propose a tree search method that is empirically Pareto-optimal.

Process Reward Models. Process reward models (PRMs) have emerged as a technique to improve the reasoning and problem-solving capabilities of LLMs. These models assign rewards to the intermediate steps of the LLM generated sequences. PRMs have been shown effective in selecting reasoning traces with a low error rate, and for providing rewards in reinforcement learning-style algorithms [Uesato et al., 2022, Polu and Sutskever, 2020, Gudibande et al., 2023]. Ma et al. [2023] applies the PRM to give rewards on the intermediate steps and guide the multi-step reasoning process. The PRM can be either trained on human labeled data [Lightman et al., 2023a] or model-labeled synthetic data [Wang et al., 2023]. In our work, we use the PRM as the reward model for selecting generated solutions, and for selecting which partial solutions to explore in tree search.

3 An Empirical Analysis of Compute-Optimal Inference for Problem-Solving

We explore the following question: *Given a fixed FLOPs budget, how should one select an optimal model size for the policy model, and an effective inference strategy to maximize performance (i.e., accuracy)?* To address this, we represent the problem-solving error rate $E(N, T)$ as a function of the number of model parameters N and the number of generated tokens T. The computational budget C is a deterministic function FLOPs(N, T), based on N and T. Our goal is to minimize E under the test-time compute constraint FLOPs$(N, T) = C$:

$$N_{opt}(C), T_{opt}(C) = \underset{N,T \text{ s.t. FLOPs}(N,T)=C}{\arg\min} E(N, T) \tag{1}$$

where $N_{opt}(C)$ and $T_{opt}(C)$ denote the optimal allocation of a computational budget C.

Here, the inference computation (FLOPs) for a fixed model can be modulated by generating more tokens with the policy model, e.g., by sampling additional candidate solutions and subsequently ranking them using a reward model. We primarily consider sampling and tree-search approaches with reranking or majority voting as the means to consume more tokens, including Greedy Search, Majority Voting, Best-of-N, Weighted Voting, and their variants on tree search methods.

3.1 Inference Strategies

3.1.1 Sampling

Greedy Search. This strategy generates tokens one at a time by selecting the highest probability token at each step, without considering future steps. It is computationally efficient but often suboptimal in terms of diversity.

Best-of-n. This strategy, also known as rejection sampling, samples multiple solutions and chooses the one with the highest score given by the reward model.

Majority Voting. In this strategy, multiple model outputs are generated, and the final answer to the problem is determined by the most frequently occurring answer in all the outputs.

Weighted Majority Voting. This strategy is a variant of majority voting in which the votes are weighted based on the score given by the reward model.

3.1.2 Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search (MCTS) has proven effective in domains such as board games where strategic decision-making is required [Silver et al., 2016, 2017, Jones, 2021]. Recent work has shown that adapting MCTS to the context of LLMs can enhance the text generation process [Zhang et al., 2023, Zhou et al., 2023, Liu et al., 2024, Choi et al., 2023, Chen et al., 2024a, Tian et al., 2024, Chen et al., 2024a]. In this context, MCTS is often paired with a value model to score and guide the exploration steps. For additional background, we provide a review of MCTS in Appendix B.

Recent work in MCTS or its variants (e.g., Tree of Thoughts [Yao et al., 2023]) mainly focus on improving the performance (e.g., accuracy) on the studied tasks. However, generic comparisons of MCTS with conventional methods like Best-of-n and Majority Voting in terms of computational budget, measured in generated tokens or processing time, are either scarce or indicating inference-time issues. For example, MCTS consumes substantially more resources, often requiring dozens of times more generated tokens than simpler methods. Specifically, a significant portion of the paths in the search tree are used to estimate and select nodes, and these paths do not necessarily become a part of the final candidate solution, although MCTS ensures that the sampled solutions comprise high-quality intermediate steps. In contrast, sampling methods generate multiple solutions in parallel and independently, and all the generated sequences are included in the candidate solutions. However, the intermediate steps in these sequences are not guaranteed to be of high quality, as there is no mechanism for pruning poor steps or exploiting promising ones.

This highlights the need for developing a new tree search method that can achieve a comparable (or better) performance as MCTS, and that is computationally less costly, just like weighted majority voting and best-of-n. This need leads to the development of our new method named Reward Balanced SEarch (REBASE), as introduced next.

3.1.3 Reward Balanced Search (REBASE)

The REBASE tree search method inherits the exploitation and pruning properties of tree search, while using the reward model alone to estimate the nodes' qualities without additional computation for estimating values by sampling children. The efficiency is achieved by constraining the total expansion width of the tree at a certain depth. REBASE balances the expansion width among the nodes at the same depth based on the rewards given by the Process Reward Model (PRM). The details are provided below:

Notations. We consider the fine-tuned LLM as a policy π_θ. Given a question q and the first k steps of a solution $x_1, \cdots, x_k$, the $(k+1)$-th step is produced by $\pi_\theta(x_{k+1}|q, x_1 \cdots x_k)$. When generating solutions using tree search, the root of the tree corresponds to the question q. The node corresponding

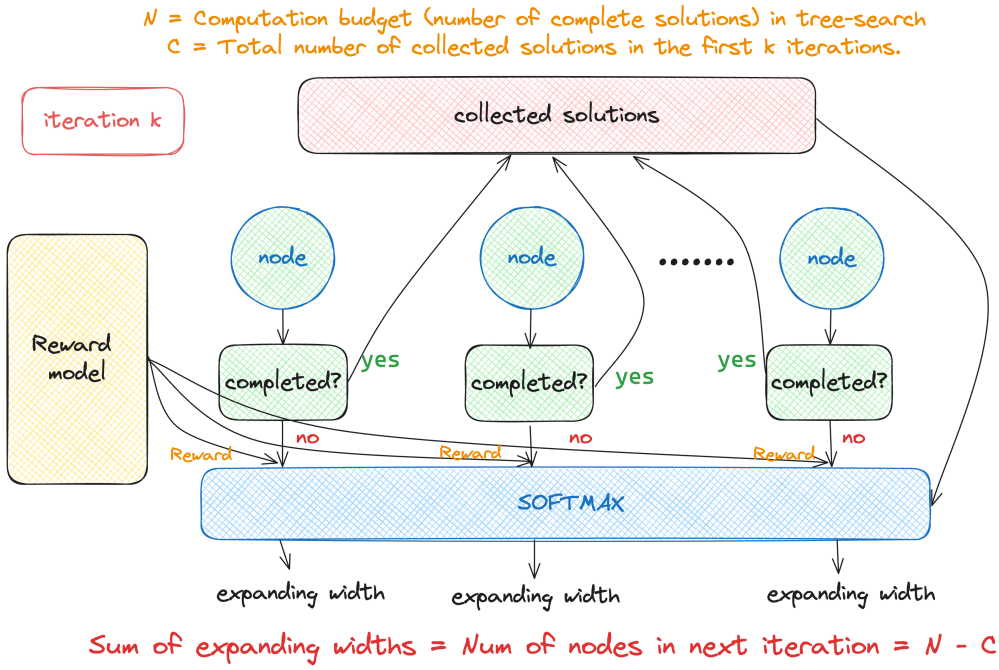

Figure 3: Illustration of one iteration of REward BAlanced SEarch (REBASE).

to x_{k+1} is the child of the node corresponding to x_k if it is sampled from $\pi_\theta(\cdot|q, x_1 \cdots, x_k)$. The reward of a node $n(x_k)$ is determined by the PRM as $R(n(x_k)) = R(q, x_1, \cdots, x_k)$.

Initialization. Given the question q, balance temperature T_b, and sampling number of solutions N, we sample N instances of the first step for the question, yielding all the nodes of depth 1 in the search tree. We set the sampling budget of depth 0 $B_0 = N$ as initialization.

Reward modeling and update. In the i-th iteration, the PRM assigns the rewards to all the nodes at depth i. After that, the algorithm examines whether the solutions up to depth i are complete. Supposing there are C_i completed solutions, we update the sampling budget using $B_i \leftarrow B_{i-1} - C_i$. If $B_i = 0$, the process ends, and we obtain N solutions.

Exploration balancing and expansion. For all of the nodes n_j with reward $R(n_j)$ in the depth i of the tree, we calculate the expansion width of the n_j as:

$$W_j = \text{Round}\left(B_i \frac{\exp\left(R(n_j)/T_b\right)}{\sum_k \exp\left(R(n_k)/T_b\right)}\right). \tag{2}$$

Then we sample W_j children for n_j for all the nodes in depth i, and start the next iteration.

3.1.4 Theoretical Analysis

Before empirically studying the scaling effects of increasing the inference-time compute budget, we present two theorems which will help us understand the experimental results later. These two theorems give an upper bound on the performance of sampling when fixing the LLM generator.

We assume the vocabulary is limited and the sequence length is constrained, thus the number of possible solutions and answers are finite. The proofs are provided in the Appendix A.

Theorem 1. *Given a test dataset $\mathcal{D}$ and a LLM π. $|\mathcal{A}|$ is the finite set of all possible answers given by LLM, the ground truth function g maps test data d to the true answer. Denote the accuracy of the LLM on this dataset with majority over N samples as $ACC_{MV}(\pi, \mathcal{D}, N)$. The accuracy of majority voting on the LLM will eventually saturate, i.e.*

$$\lim_{N\to\infty} ACC_{MV}(\pi, \mathcal{D}, N) = \frac{\sum_{d\in\mathcal{D}} \mathbb{I}\left((g(d) = \arg\max_{a\in\mathcal{A}} \pi(a|d)\right)}{|\mathcal{D}|}. \tag{3}$$

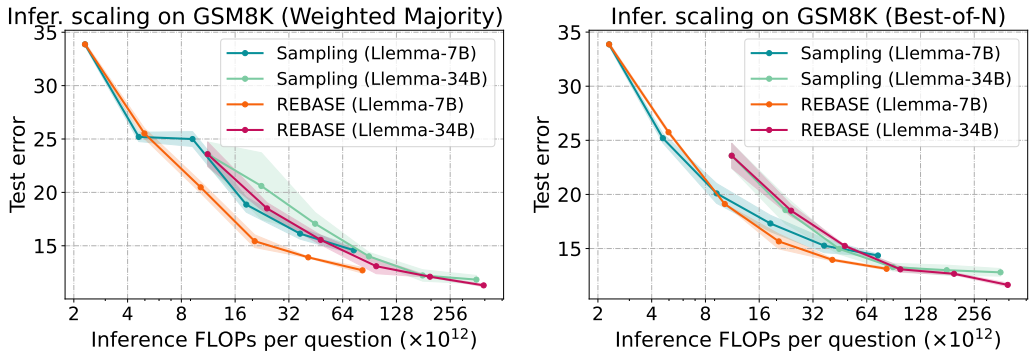

Figure 4: **The inference computation scaling comparisons across different model sizes**. The left/right panel shows the GSM8K problem-solving error rate on GSM8K based on Weighted Mjority/Best-of-N.

where $\pi(x|d)$ denotes the probability that the LLM answers x given input d and $\mathbb{I}$ is the indicator function.

Theorem 2. *Assume the reward model assigns an expected reward of $R(a)$ to $a \in \mathcal{A}$ among the different solutions generated by LLM that yields a. Given a test dataset $\mathcal{D}$ and a LLM π. $|\mathcal{A}|$ is the finite set of all possible answers given by LLM, the ground truth function g maps test data d to the true answer. Denote the accuracy of the LLM on this dataset with weighted majority over N samples as $ACC_{WV}(\pi, \mathcal{D}, N, R)$. The accuracy of weighted majority voting on the LLM will eventually saturate, i.e.*

$$\lim_{N\to\infty} ACC_{WV}(\pi, \mathcal{D}, N, R) = \frac{\sum_{d\in\mathcal{D}} \mathbb{I}\left((g(d) = \arg\max_{a\in\mathcal{A}} R(a)\pi(a|d)\right)}{|\mathcal{D}|}. \tag{4}$$

where $\pi(x|d)$ denotes the probability that the LLM answers x given input d and $\mathbb{I}$ denotes the indicator function.

Theorem 2 shows that as long as the reward model assigns higher rewards than the policy for correct answers versus other answers in expectation, the upper bound of Weighted Majority Voting will be higher than Majority Voting since $\mathbb{I}\left((g(d) = \arg\max_{a\in\mathcal{A}} R(a)\pi(a|d)\right) > \mathbb{I}\left((g(d) = \arg\max_{a\in\mathcal{A}} \pi(a|d)\right)$. We put the figures comparing BoN and Weighted Majority Voting in the main paper and leave the Majority Voting figures in Appendix D since Majority Voting is dominated by Weighted Majority Voting.

4 Experiments

4.1 Setup

Datasets. We conduct experiments on two mathematical problem-solving datasets to investigate the scaling effects of computation and our REBASE method for both challenging and simpler problems. Specifically, MATH [Hendrycks et al., 2021a] and GSM8K[Cobbe et al., 2021b] are datasets containing high school mathematics competition-level problems and grade-school level mathematical reasoning problems, respectively. Following [Lightman et al., 2023b, Wang et al., 2024, Sun et al., 2024], we use the MATH500 subset as our test set.

Generators. We use Llemma-7B and Llemma-34B [Azerbayev et al., 2024] as our base models and finetune them on the MetaMath dataset [Yu et al., 2024] using full parameter supervised fine-tuning (Full-SFT), The detailed finetuning configuration is given in the Appendix. Additionally, we test the Mistral-7B model to expand our findings across different models.

Reward Model. All of the experiments use the same Llemma-34B reward model, which we finetuned on the synthetic process reward modeling dataset, Math-Shepherd [Wang et al., 2024]. We added a reward head to make the model, enabling it to output a scalar reward at the end of each step.

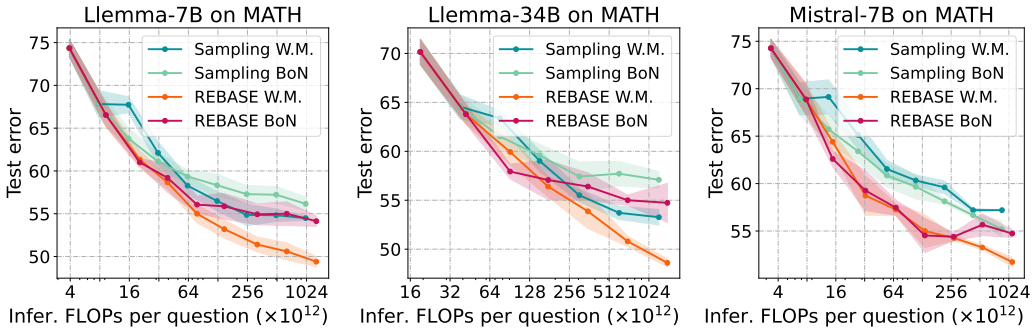

Figure 5: **The inference computation scaling laws** of different models for the problem-solving error rate on **MATH500** test set. The tested models are Llemma-7B (left), Llemma-34B (middle), & Mistral-7B (right). In the legend, W.M. and BoN refer to Weighted Majority and Best-of-N, respectively.

Inference Configuration. For the MATH dataset, we sample 1, 2, 4, 8, 16, 32, 64, 128, and 256 solutions for the 7B models, and 1 to 64 solutions for the 34B Llemma model. For the GSM8K dataset, we sample 1 to 32 solutions, as it is relatively easier. We use sampling and REBASE to generate these samples and select the answer through Best-of-N, Majority Voting, and Weighted Voting. Each configuration is run multiple times to calculate the mean and variance, thereby mitigating the randomness and ensuring the reliability of our conclusions.

4.2 Main Results of Compute-Optimal Inference

In order to compare the compute budgets of 7B and 34B models, we plot the figures with the number of FLOPs used per question during inference. We compute the inference FLOPs based on the standard formula from [Kaplan et al., 2020].

Llemma-7B model achieves competitive accuracy to Llemma-34B model with lower compute budget. Figures 1 and 4 show the curves of error rates versus total number of inference FLOPs per question. Inference methods with different model sizes are plotted in the same diagram. We found that Llemma-7B costs approximately 2x less total FLOPs than Llemma-34B under the same method (Sampling, MCTS, REBASE) and task (MATH, GSM8K) while achieving competitive accuracy. This result suggests that, with the same training dataset and model family, training and inference with a smaller model could be more favorable in terms of compute budget if multiple sampling or search methods are employed.

All inference configurations will saturate eventually. This is expected as Theorem 1 and Theorem 2 show. Also illustrated in Figures 5 and 6, the slope of the erro rate curves start large, then decreases and the curves finally become nearly flat as the number of samples scales, showing the effect of saturation.

Scaling law of compute-optimal inference. The findings in our experiments are consistent with the Theorem 1 and 2, After a threshold the accruacy of sampling more solutions saturate, we should scale the model size. We interpolate the smoothed test error rate curve in Figure 1 and Figure 4, and fit power laws to estimate the optimal model size N and number of generated tokens T for any given amount of compute. We obtained a relationship $N_{opt} \propto C^a$ and $T_{opt} \propto C^b$, where $a = 1.0$ and $b = 0.0$ for both sampling-based weighted voting and our tree-search method REBASE. Our fitted curves indicate that the optimal inference strategy is invariant to the amount of compute (e.g., re-ranking with 32 sampled solutions or REBASE tree search with a compute budget of 64 for MATH), and the optimal model size grows linearly with the increased compute budget.

4.3 Comparing REBASE to Other Baselines

REBASE is Pareto-optimal. While MCTS undeperforms Sampling (Fig. 1), from Fig. 1, 4, 5, and 6, we found that REBASE consistently outperforms the Sampling method in all settings, when

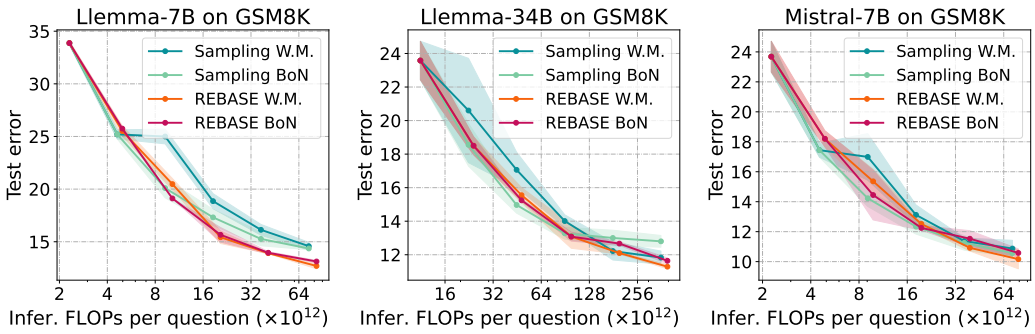

Figure 6: **The inference computation scaling laws** of different models for the problem-solving error rate on **GSM8K** test set. The tested models are Llemma-7B (left), Llemma-34B (middle), & Mistral-7B (right). In the legend, W.M. and BoN refer to Weighted Majority and Best-of-N, respectively.

Table 1: REBASE with lower compute budget has competitive accuracy against Sampling with higher compute budget. We use weighted voting to aggreagte the candidate solutions in both Sampling and REBASE.

	# SAMPLES	FLOPS	MATH500
MISTRAL-7B			
SAMPLING	256	8.7×10^{14}	42.8
REBASE	32	$\mathbf{1.36 \times 10^{14}}$	**45.0**
LLEMMA-7B			
SAMPLING	256	10.0×10^{14}	45.5
REBASE	32	$\mathbf{1.48 \times 10^{14}}$	**46.8**
LLEMMA-34B			
SAMPLING	64	12.1×10^{14}	46.7
REBASE	32	$\mathbf{7.08 \times 10^{14}}$	**49.2**

fixing the model and the evaluation task. Table 1 shows that REBASE can achieve competitive accuracy with even a lower compute budget than the sampling method. This finding is novel, and differs from previous tree search works which typically improve the performance at the cost of higher computational expense compared to sampling [Chen et al., 2024a, Xie et al., 2023]. Table 2 shows that given the same compute budget (sampling 32 solutions for the 7B model and 8 solutions for 34B model), using REBASE yields higher accuray than sampling.

Weaker models gain more from Tree Search. From Fig. 2, we saw that compared with sampling, Mistral-7B, Llemma-7B, Llemma-34B increase 5.3%, 3.3%, 2.6% in MATH and 0.7%, 1.9%, 0.9% in GSM8K. The order of accuracy increase is inversely related to the model's corresponding greedy search on those datasets. This suggests that weaker models, as indicated by their lower greedy search accuracy, benefit more from tree search methods like REBASE.

REBASE saturates later than sampling with higher accuray. From Figure 5 and Figure 6, we observe that both sampling and REBASE saturate early in GSM8K and relatively late in MATH, which we attribute to the difference of the difficulty level. This can be explained through the LLM may assign high probability to the true answer in easy problems than those of harder problem, as suggested by Theorem 1 and 2 with their proofs A. On MATH (Figure 5), we see that REBASE finally saturates with a higher accuracy than sampling. We hypothesize the reason is that REBASE samples the truth answer with higher probability than sampling. And as demonstrated by Theorem 1 and 2, the upper bound becomes higher.

Table 2: Accuracy of diffrent inference configurations under a specific compute budget. MV, BoN and WV denote Majority Voting, Best-of-N and Weighted Voting, respectively.

	# SAMPLES	MATH FLOPS	GSM8K FLOPS	MATH500	GSM8K
			MISTRAL-7B		
GREEDY	1	3.4×10^{12}	2.3×10^{12}	28.6	77.9
SAMPLING + MV	32	109.2×10^{12}	72.6×10^{12}	36.1	85.7
SAMPLING + BON	32	109.2×10^{12}	72.6×10^{12}	40.3	89.4
SAMPLING + WV	32	109.2×10^{12}	72.6×10^{12}	39.7	89.1
REBASE + MV	32	136.2×10^{12}	78.9×10^{12}	44.1	88.8
REBASE + BON	32	136.2×10^{12}	78.9×10^{12}	**45.4**	89.4
REBASE + WV	32	136.2×10^{12}	78.9×10^{12}	45.0	**89.8**
			LLEMMA-7B		
GREEDY	1	3.92×10^{12}	2.3×10^{12}	30.0	68.5
SAMPLING + MV	32	125.4×10^{12}	73.9×10^{12}	41.0	80.0
SAMPLING + BON	32	125.4×10^{12}	73.9×10^{12}	41.7	85.6
SAMPLING + WV	32	125.4×10^{12}	73.9×10^{12}	43.5	85.4
REBASE + MV	32	148.0×10^{12}	82.6×10^{12}	46.1	86.1
REBASE + BON	32	148.0×10^{12}	82.6×10^{12}	44.1	86.9
REBASE + WV	32	148.0×10^{12}	82.6×10^{12}	**46.8**	**87.3**
			LLEMMA-34B		
GREEDY	1	19.0×10^{12}	11.2×10^{12}	33.0	78.4
SAMPLING + MV	8	152.3×10^{12}	89.7×10^{12}	39.9	84.3
SAMPLING + BON	8	152.3×10^{12}	89.7×10^{12}	40.4	86.7
SAMPLING + WV	8	152.3×10^{12}	89.7×10^{12}	41.0	86.0
REBASE + MV	8	176.8×10^{12}	98.7×10^{12}	**43.9**	86.1
REBASE + BON	8	176.8×10^{12}	98.7×10^{12}	43.6	**86.9**
REBASE + WV	8	176.8×10^{12}	98.7×10^{12}	42.9	**86.9**

5 Conclusion & Limitations

In this work, we have conducted a comprehensive empirical analysis of compute-optimal inference for problem-solving with language models. Our study has revealed several key findings. First, with an optimal inference configuration, a small language model can achieve competitive accuracy to a $4\times$ larger model while using approximately $2\times$ less total FLOPs under the same inference method (Sampling, MCTS, REBASE) and task (MATH, GSM8K), suggesting that training and inference with smaller models could be more favorable in terms of compute budget when combined with multiple sampling or search strategies. Second, our new REBASE tree-search method consistently outperforms sampling (and MCTS) across all settings, models, and tasks, achieving competitive accuracy with even lower compute budget compared to sampling. Our findings highlight the potential of deploying smaller models equipped with more sophisticated inference strategies like REBASE to enhance problem-solving accuracy while maintaining computational efficiency.

Limitations First, our experiments focused specifically on mathematical problem-solving tasks, and the generalizability of our findings to other domains remains to be explored. Second, we only investigated a limited range of model scales, primarily focusing on 7B and 34B models. Future research could extend our analysis to a wider range of model sizes to gain a more comprehensive understanding of the scaling laws for compute-optimal inference. Finally, our experiments mainly utilized the MetaMath dataset for training the models. It would be valuable to explore the impact of different training datasets on the performance and efficiency of compute-optimal inference strategies for mathematical problem-solving.

References

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*, 2023.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics, 2024.

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024. URL `https://openai.com/research/video-generation-models-as-world-simulators`.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. Alphamath almost zero: process supervision without process, 2024a.

Ziru Chen, Michael White, Raymond Mooney, Ali Payani, Yu Su, and Huan Sun. When is tree search useful for llm planning? it depends on the discriminator. *arXiv preprint arXiv:2402.10890*, 2024b.

Sehyun Choi, Tianqing Fang, Zhaowei Wang, and Yangqiu Song. Kcts: Knowledge-constrained tree search decoding with token-level hallucination detection, 2023.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. PaLM: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021a.

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

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR, 2023.

Alex Graves. Sequence transduction with recurrent neural networks, 2012.

Arnav Gudibande, Eric Wallace, Charlie Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. The false promise of imitating proprietary llms. *arXiv preprint arXiv:2305.15717*, 2023.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, et al. Measuring coding challenge competence with apps. *arXiv preprint arXiv:2105.09938*, 2021a.

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

Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B. Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Patwary, Mostofa Ali, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Andy L Jones. Scaling scaling laws with board games. *arXiv preprint arXiv:2104.03113*, 2021.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *arXiv preprint arXiv:2206.14858*, 2022.

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

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023b.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, 2017.

Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. Don't throw away your value model! generating more preferable text with value-guided monte-carlo tree search decoding, 2024.

Qianli Ma, Haotian Zhou, Tingkai Liu, Jianbo Yuan, Pengfei Liu, Yang You, and Hongxia Yang. Let's reward step by step: Step-level reward model as the navigators for reasoning, 2023.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

OpenAI. Gpt-4 technical report, 2023.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.

Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*, 2020.

Jonathan S Rosenfeld, Amir Rosenfeld, Yonatan Belinkov, and Nir Shavit. A constructive prediction of the generalization error across scales. *arXiv preprint arXiv:1909.12673*, 2019.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Zhiqing Sun, Longhui Yu, Yikang Shen, Weiyang Liu, Yiming Yang, Sean Welleck, and Chuang Gan. Easy-to-hard generalization: Scalable alignment beyond human supervision. *arXiv preprint arXiv:2403.09472*, 2024.

Virginia Teller. Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 2000.

Ye Tian, Baolin Peng, Linfeng Song, Lifeng Jin, Dian Yu, Haitao Mi, and Dong Yu. Toward self-improvement of llms via imagination, searching, and criticizing. *arXiv preprint arXiv:2404.12253*, 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.

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

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *International Conference on Learning Representations (ICLR 2023)*, 2022a.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*, 2022b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS*, 2022.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. Self-evaluation guided beam search for reasoning, 2023.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2(3):5, 2022.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models, 2024.

Shun Zhang, Zhenfang Chen, Yikang Shen, Mingyu Ding, Joshua B. Tenenbaum, and Chuang Gan. Planning with large language models for code generation, 2023.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models, 2023.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*, 2022.

A Proofs of Theorem 1 and 2

A.1 Proof of Theorem 1

Proof. Suppose the possible answers of the LLM are $x_1, x_2, x_3, \cdots, x_{|\mathcal{A}|}$, with $\pi(x_1|d) > \pi(x_2|d) > \cdots > \pi(x_{|\mathcal{A}|}|d)$. After sampling N solutions from the LLM, we denote the occurence of x_i as $f(x_i)$, the probability that x_1 is not the most frequent output is

$$P(f(x_1) \neq \arg\max_x f(x)) \tag{5}$$

With Union bound, we get

$$\begin{aligned}
&P(x_1 \neq \arg\max_x f(x)) && (6)\\
\leq &\sum_{i=2}^{|\mathcal{A}|} P(f(x_1) \leq f(x_i)) && (7)\\
\leq &|\mathcal{A}| P(f(x_1) \leq f(x_2)) && (8)\\
= &|\mathcal{A}| \left(1 - P\left(f(x_1) \geq f(x_2)\right)\right) && (9)\\
\leq &|\mathcal{A}| \left(1 - P\left(f(x_1) \geq \frac{\pi(x_1|d) + \pi(x_2|d)}{2} N\right) P\left(f(x_2) \leq \frac{\pi(x_1|d) + \pi(x_2|d)}{2} N\right)\right) && (10)\\
\leq &|\mathcal{A}| \left(1 - \left(1 - e^{-\frac{\delta_1^2}{2}\pi(x_1|d)N}\right)\left(1 - e^{-\frac{\delta_2^2}{2+\delta_2}\pi(x_2|d)N}\right)\right) && (11)\\
\leq &|\mathcal{A}| C^N \quad \text{for some } C < 1. && (12)
\end{aligned}$$

Where (11) is by Chernoff Bound, $\delta_1 = \frac{\pi(x_1|d) - \pi(x_2|d)}{2\pi(x_1|d)}$ and $\delta_2 = \frac{\pi(x_1|d) - \pi(x_2|d)}{2\pi(x_2|d)}$. As $N \to \infty$, we have

$$f(x) = \begin{cases} M(x|N) = 1 & \text{if } x = \arg\max_{a \in \mathcal{A}} \pi(a|d) \\ M(x|N) = 0 & \text{otherwise .} \end{cases} \tag{13}$$

Where $M(x|N)$ denotes the probability that majority voting over N sampled solutions gives x. The proof of original theorem is automatically completed by (13). □

A.2 Proof of Theorem 2

Proof. The proof is similar to the Theorem 1, We rank $x_1, x_2, \cdots, x_{|\mathcal{A}|}$ with $R(x_1)f(x_1) > \cdots > R(x_{|\mathcal{A}|}f(x_{|\mathcal{A}|})$. Denotes $w(x_i)$ as the the total weights of answer x_i after sampling N solutions. As $N \to \infty$, $w(x_i) \to R(x_i)f(x_i)$. Same as proof in theorem 1, we have

$$\begin{aligned}
&P(x_1 \neq \arg\max_x f(x)) && (14)\\
\leq &|\mathcal{A}| P(w(x_1) \leq w(x_2)) && (15)\\
= &|\mathcal{A}| \left(1 - P\left(w(x_1) \geq w(x_2)\right)\right) && (16)\\
\leq &|\mathcal{A}| \left(1 - P\left(w(x_1) \geq \frac{v(x_1) + v(x_2)}{2} N\right) P\left(w(x_2) \leq \frac{v(x_1) + v(x_2)}{2} N\right)\right). && (17)
\end{aligned}$$

Where $v(x) = R(x)\pi(x|d)$, the remaining proof completely follows Theorem 1. □

B MCTS Details

The MCTS process can be represented as the following steps:

Selection The process begins at the root node. Here, the algorithm recursively selects the child node that offers the highest Upper Confidence Bound applied to Trees (UCT) value, continuing until a node is reached that has not been expanded. The UCT is calculated using the formula

$$UCT(s) = Q(s) + C\sqrt{\frac{\ln\left(N(Parent(s))\right)}{N(s)}}. \tag{18}$$

Table 3: Fine-tuning Hyper-parameters: LR refers to the learning rate, BS refers to the batch size. Llemma-7B and LLemma-34B are the generators we use in our experiments, RM is short for Reward Model.

Model	# Epoch	Dataset	BS	LR	Max Seq Length	Dtype
Llemma-7B	1	MetaMath	128	8E-6	1024	FP32
Llemma-34B	1	MetaMath	128	8E-6	768	FP32
Llemma-34B RM	2	Math-Shepherd	128	1E-5	768	BF16

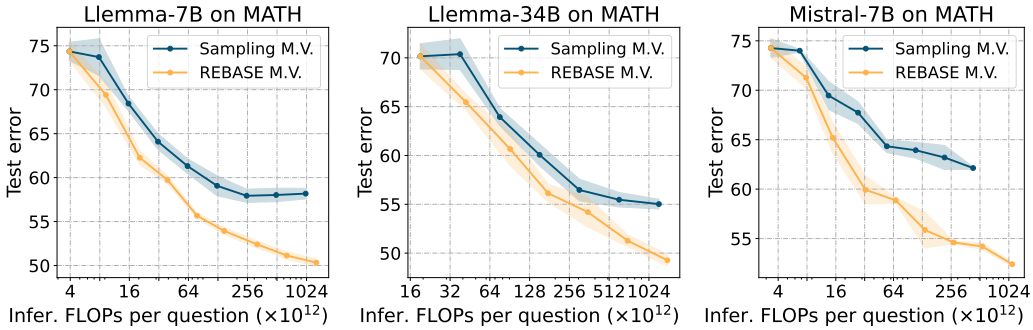

Figure 7: **The inference computation scaling laws** of different models for the problem-solving error rate on **MATH** test set. The tested models are Llemma-7B (left), Llemma-34B (middle), & Mistral-7B (right). In the legend, M.V. refer to Majority Voting.

Where $Q(s)$ represents the quality score of node s, $N(s)$ is the number of visits to node s, and C is a constant determining the level of exploration.

Expansion and evaluation Upon reaching a non-terminal node s, the node is expanded by generating multiple child nodes. Each child node c is then evaluated using a value function $V(c)$, which predicts the potential quality of continuing the sequence from node c.

Backpropagation After evaluation, the algorithm updates the UCT values and the visit counts for all nodes along the path from the selected node back to the root. For any node n in this path, the updates are made as follows:

$$N(n) \leftarrow N(n) + 1,$$
$$Q(n) \leftarrow \frac{(N(n)-1)\,Q(n) + V(s)}{N(n)}.$$

C Hyper-parameters

Finetuning We put all the hyperparameters of fine-tuned models in the table 3. We preprocess the MetaMath Dataset to make the solutions in a stepwise format.

Inference For all the inference strategies, the temperature of the LLM is set to 1.0. Max tokens for the output is 1024 and max tokens for one step is 256. For REBASE, we chose the balance temperature (the softmax temperature in the REBASE algorithm) as $T_b = 0.1$. For MCTS, we set C in the UCT value as 1 and we expand $4, 8, 16$ children for the root, 2 children for other selected nodes with total 32, 64, 128 expansions respectively.

D Supplementary Figures

In the main part of paper, there isn't enough space for showing the scaling effects of Majority Voting, we append the figures about Majority Voting and Majority Voting v.s. Weighted Majority Voting

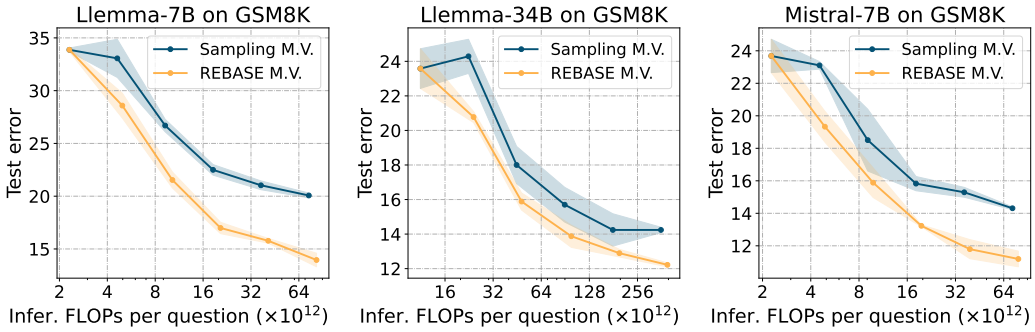

Figure 8: **The inference computation scaling laws** of different models for the problem-solving error rate on **GSM8K** test set. The tested models are Llemma-7B (left), Llemma-34B (middle), & Mistral-7B (right). In the legend, M.V. refer to Majority Voting.

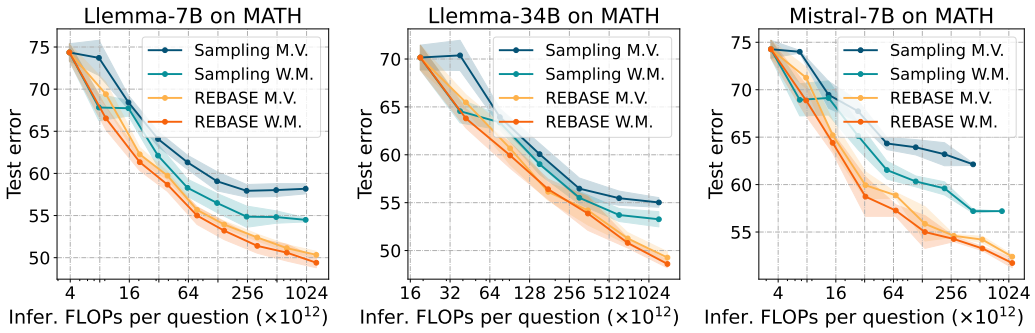

Figure 9: **The inference computation scaling laws** of different models for the problem-solving error rate on **MATH** test set. The tested models are Llemma-7B (left), Llemma-34B (middle), & Mistral-7B (right). In the legend, M.V. and W.M. refer to Majority Voting and Weighted Majority, respectively.

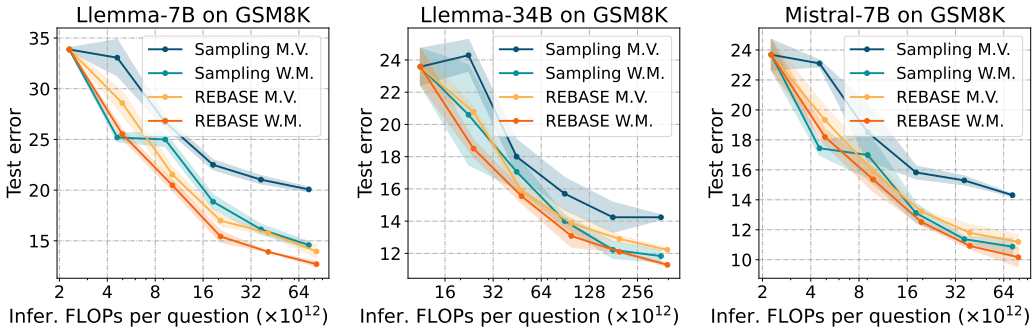

Figure 10: **The inference computation scaling laws** of different models for the problem-solving error rate on **GSM8K** test set. The tested models are Llemma-7B (left), Llemma-34B (middle), & Mistral-7B (right). In the legend, M.V. and W.M. refer to Majority Voting and Weighted Majority, respectively.

(Fig. 7, 8 ,9, 10) in this appendix. The experiments show that although the gap between Majority Voting and Weighted Majority Voting on sampling is huge. This gap becomes much smaller if we apply REBASE. This phenomenon can be caused by the selection ability of tree search like REBASE. Once REBASE already samples solutions with high rewards, conducing weighted majority voting gains less since the sampled solutions may all have relatively high and stable rewards compared with those of sampling.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: In Abstract and Introduction, we claim that we investigate the compute-optimal inference: designing models and inference strategies that optimally trade off additional inference-time compute for improved performance.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The discussion is in the last section of the main paper.

   Guidelines:

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

3. **Theory Assumptions and Proofs**

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

Answer: [Yes]

Justification: It's in Appendix.

Guidelines:

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

4. **Experimental Result Reproducibility**

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

Answer: [Yes]

Justification: We introduce our method in Section 3 and the hyperparameters are introduced in the Appendix.

Guidelines:

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

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

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

Answer: [Yes]

Justification: We only used open-source models in this work. The code will be released.

Guidelines:

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

6. **Experimental Setting/Details**

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

Answer: [Yes]

Justification: We used the standard training and test splits or MATH and GSM8K and report the hyperparameters in the appendix.

Guidelines:

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

7. **Experiment Statistical Significance**

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

Answer: [Yes]

Justification: The error bars are included in our figures.

Guidelines:

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

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

8. **Experiments Compute Resources**

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

   Answer: [Yes]

   Justification: All the experiments are conducted on $8\times$ H100 GPUs.

   Guidelines:

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

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: Yes, we conform with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader Impacts**

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

    Answer: [NA]

    Justification: We do not find significant positive societal impacts and negative societal impacts of our work.

    Guidelines:

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

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

11. **Safeguards**

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

    Answer: [NA]

    Justification:

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We use the proper citations.

    Guidelines:

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

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

13. **New Assets**

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

    Answer: [Yes]

    Justification: We use the proper citations.

    Guidelines:

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

14. **Crowdsourcing and Research with Human Subjects**

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

    Answer: [NA]

    Justification: No crowdsourcing experiments are used.

    Guidelines:

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

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

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

    Answer: [NA]

    Justification: No human subjects are involved.

    Guidelines:

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