INFERENCE SCALING LAWS: AN EMPIRICAL ANALYSIS OF COMPUTE-OPTIMAL INFERENCE FOR LLM PROBLEM-SOLVING

Anonymous authors
Paper under double-blind review

ABSTRACT

While the scaling laws of large language models (LLMs) training have been extensively studied, optimal inference configurations of LLMs remain underexplored. We study *inference scaling laws* and *compute-optimal inference*, focusing on the trade-offs between model sizes and generating additional tokens with different inference strategies. As a first step towards understanding and designing compute-optimal inference methods, we studied cost-performance trade-offs for inference strategies such as greedy search, majority voting, best-of-n, weighted voting, and two different tree search algorithms, using different model sizes and compute budgets. Our findings indicate smaller models (e.g., Llemma-7B) can outperform larger models given the same computation budgets, and that smaller models paired with advanced inference algorithms yield Pareto-optimal cost-performance trade-offs. For instance, the Llemma-7B model, equipped with our novel tree search algorithm, consistently outperforms Llemma-34B with standard majority voting on the MATH benchmark across all FLOPs budgets. We hope these findings contribute to a broader understanding of inference scaling laws for LLMs.

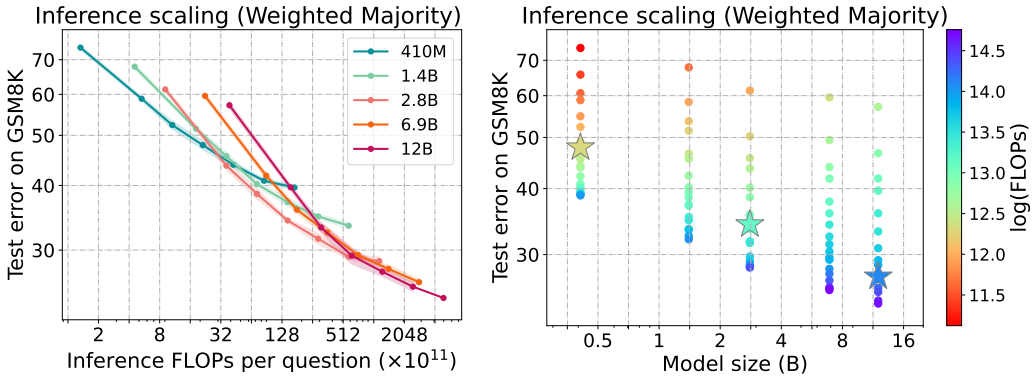

Figure 1: **Inference scaling laws** exhibited for Pythia (Biderman et al., 2023) models and **GSM8K** test error. We evaluate the error rate (lower is better) of models using various sizes and numbers of sampled solutions for weighted majority voting. *Left:* the error rate for each model size decreases steadily as inference-compute increases, and converges at the end. *Right:* the optimal model size (shown as stars for 2^{41}, 2^{44}, and 2^{47} FLOPs) varies based on the inference-time compute budget. For instance, smaller models are compute-optimal at 2^{41} and 2^{44} FLOPs. Both axes are $\log$ scale.

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 & 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 as 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; Pfau et al., 2024). The computational cost of these techniques 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 require much more compute 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 (Wang et al., 2022a; Li et al., 2023)) 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 size for the language model and an effective inference strategy (e.g., greedy search, majority voting, best-of-n, weighted voting, and their tree-search variants) to maximize performance (i.e., accuracy) with a given compute budget. We control the inference computation (FLOPs) of a fixed model by generating more tokens through the language model[1], sampling further candidate solutions, and ranking them with a reward model. We analyze the performance of fine-tuned models of various sizes given different inference FLOPs on mathematical reasoning benchmarks (e.g., GSM8K test set (Cobbe et al., 2021a) and MATH500 test set (Hendrycks et al., 2021b; Lightman et al., 2023b)). Our experiments cover several model families, including general-purpose LLMs (e.g., Pythia (Biderman et al., 2023) & Mistral (Jiang et al., 2023)) as well as math-specialized ones (e.g., Llemma (Azerbayev et al., 2023)).

Our results on Pythia (Fig. 1) illustrate how performance scales with increased inference compute across various model sizes. Typically, increasing the compute budget leads to higher accuracy until the accuracy reaches saturation. As the compute budget increases, smaller models initially perform better than larger ones, but once the accuracy of the smaller models saturates, the larger models have favorable performance. The right panel of Figure 1 demonstrates that the optimal model size for inference varies with different levels of computation. However, in real-world deployment, the available computation is typically much lower than the point where the accuracy of relatively small models saturates and larger models begin to show their advantage (as shown in Figure 4, where the 7B model outperforms the 34B model until 128 Llemma 7B solutions are sampled). This indicates that relatively smaller models could be more compute-optimal for inference.

We analyse the asymptotic behaviour of sampling and voting-based inference strategies, showing their convergence upper bound and rate of convergence. Given a dataset, the accuracy of the language model will ultimately saturate to a fixed limit which is determined by the output probabilities assigned by the model, exhibiting exponential convergence speed through sampling and voting. This implies that, without an oracle verifier, simple strategies like sampling cannot achieve perfect accuracy even with an infinite number of samples, leading to diminishing returns. Therefore, this highlights the necessity for more sophisticated inference algorithms.

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 effective votes. To address this issue, we propose a novel tree search algorithm, *REward BAlanced SEarch* (REBASE), which pairs well with weighted voting and achieves a Pareto-optimal trade-off between accuracy and inference compute. The key idea of REBASE is to use a node-quality reward to control node expansion, which eliminates the need for explicit rollouts while ensuring enough candidate solutions for voting.

[1]Following Uesato et al. (2022), we refer to the main language model generating outputs as the *policy* model. It can be paired with a *reward* model, which scores outputs from the policy model to facilitate inference.

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 $2\times$ less FLOPs when evaluating on MATH500 (Fig. 4) or GSM8K (Fig. 5). Moreover, Llemma-7B with REBASE outperforms Llemma-34B with standard majority voting across *all* compute budgets. Our results show the value of using smaller models with advanced inference-time algorithms, and the benefits of new algorithms for achieving better returns on inference-time compute.

Our contributions are summarized as follows:

- We explore new inference scaling laws by evaluating the performance of various model sizes under a fixed inference strategy. We show that smaller models can outperform larger ones under the same compute budget by increasing the number of samples.
- We provide new theoretical analysis of the scaling behavior of voting methods, presenting convergence bounds and rates. Our analysis shows performance limits and diminishing returns from sampling, pointing to the need for more sophisticated inference algorithms.
- We formulate a new compute-optimal inference problem and propose a novel tree search algorithm, REBASE, which is compute-optimal compared to widely-used sampling and MCTS methods. Our results show benefits of using smaller models with advanced inference algorithms, and new algorithms for achieving better cost-performance tradeoffs.

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 training a verifier for evaluating and ranking solutions. Subsequent research (e.g., Lewkowycz et al. (2022)) has shown the performance benefits of inference-time techniques such as majority voting (Wang et al., 2022a) and weighted majority voting (Li et al., 2023). We choose problem solving in mathematics as the task to study compute-optimal strategies since it allows us to accurately evaluate problem solving ability.

Inference Strategies of LLM Problem Solving. A variety of inference 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 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 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 resulting cost-performance trade-offs 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 & Sutskever, 2020; Gudibande et al., 2023). Ma et al. (2023) applies a 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.

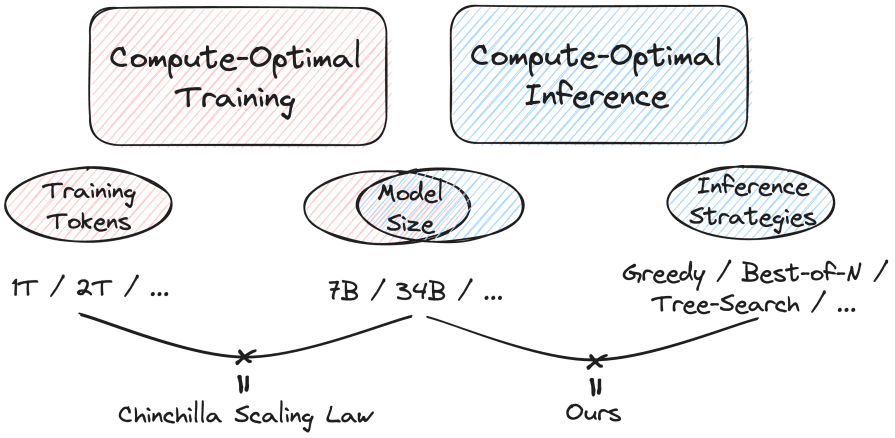

Figure 2: **Illustration of compute-optimal scaling laws in training and inference.** The Chinchilla scaling law (Hoffmann et al., 2022) shows how to choose a model size and number of training tokens under a training-compute budget, while our work shows how to choose a model size and an inference strategy under an inference-compute budget.

3 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)?* We are the first to formulate this problem and study the associated inference-time scaling laws, setting our work apart from previous scaling law studies (Fig. 2).

To address this, we represent the problem-solving error rate $E(N, T; \mathcal{S})$ as a function of the number of model parameters N, the number of generated tokens T and the inference strategy $\mathcal{S}$. The computational budget C is a deterministic function $\text{FLOPs}(N, T; \mathcal{S})$, based on N and T. Our goal is to minimize E under the test-time compute constraint $\text{FLOPs}(N, T, \mathcal{S}) = C$:

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

where $N_{\text{opt}}(C)$ and $T_{\text{opt}}(C)$ 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 and an inference strategy, e.g., sampling additional candidate solutions and subsequently ranking them using a reward model. As the inference strategies, we primarily consider sampling and tree-search approaches paired with re-ranking or majority voting. This includes greedy search, majority voting, best-of-n, weighted voting, and their tree-search variants.

3.1 Inference Strategies

We consider the following inference strategies which are popularly used in practice:

- **Greedy search.** This strategy generates tokens one at a time by selecting the highest probability token at each step. It is computationally efficient but often suboptimal in terms of diversity.
- **Best-of-n.** This strategy, also known as rejection sampling, generates a set of candidates and chooses the one with the highest score given by the reward model.
- **Majority voting.** In this strategy, a set of candidates 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 candidates are weighted based on the scores given by the reward model.

We say a strategy is **sampling-based** if it uses a standard autoregressive sampling algorithm (e.g., temperature sampling) to generate the candidate set (greedy search is separate, in that it only has a single deterministic candidate). A **tree-search** variant uses a tree search to generate the candidate set. Before discussing tree-search methods, we analyze sampling-based voting below.

Theoretical analysis of sampling-based voting. We present theoretical results on the asymptotic behavior of voting-based strategies given infinite compute in Theorems 1 & 2. Informally, we show that the accuracy of standard/weighted majority voting converges with infinite samples, and the limit only depends on the distribution modeled by the language model (and the reward model). This theoretical finding is also aligned with our empirical findings shown in Sec. 4.2, which show saturation at high sampling budgets. The proofs are presented in Appendix A.

Notations and assumptions. Let $\mathcal{V}$ be a *finite* vocabulary and $\mathcal{V}^*$ its Kleene closure, i.e., the set of all strings. Given a problem x, we say a language model answers y to this problem if the model outputs rey where $r \in \mathcal{V}^*$ can be any "reasoning path" and $\mathrm{e} \in \mathcal{V}$ denotes a special token that marks the end of reasoning. We further assume that the answer string is always shorter than L tokens, i.e., $|y| \leq L$ for some fixed $L \in \mathbb{N}^*$ where $|y|$ denotes the length of y. For a language model π, denote by $\pi(v|w)$ the probability of generating v given input (prompt) w. For a reward model ρ, denote by $\rho(v)$ the score it assigns to the string v. We use $\mathbb{I}$ to denote the indicator function.

Theorem 1. *Consider a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^m$ where x_i and y_i denotes input and true answer, respectively. For a language model π, denote by $\mathrm{acc}_n^{\mathrm{MV}}(\mathcal{D}; \pi)$ the accuracy on $\mathcal{D}$ using majority voting with n samples. Following the notations and assumptions defined above, we have:*

$$\lim_{n \to +\infty} \mathrm{acc}_n^{\mathrm{MV}}(\mathcal{D}; \pi) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{I}\left[y_i = \arg\max_{|y| \leq L} \sum_{r \in \mathcal{V}^*} \pi(rey|x_i) \right] \textit{(almost surely);}$$

$$\textit{and} \quad \mathbb{E}\left[\mathrm{acc}_n^{\mathrm{MV}}(\mathcal{D}; \pi)\right] = \frac{1}{m} \sum_{i=1}^{m} \mathbb{I}\left[y_i = \arg\max_{|y| \leq L} \sum_{r \in \mathcal{V}^*} \pi(rey|x_i) \right] - \mathcal{O}(c^{-n})$$

for some constant $c > 1$.

Theorem 2. *Consider a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^m$. For a language model π and a reward model ρ, denote by $\mathrm{acc}_n^{\mathrm{WV}}(\mathcal{D}; \pi, \rho)$ the accuracy on $\mathcal{D}$ using weighted majority voting with n samples. Following the notations and assumptions defined above, we have:*

$$\lim_{n \to +\infty} \mathrm{acc}_n^{\mathrm{WV}}(\mathcal{D}; \pi, \rho) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{I}\left[y_i = \arg\max_{|y| \leq L} \sum_{r \in \mathcal{V}^*} \pi(rey|x_i)\rho(x_i rey) \right] \textit{(almost surely);}$$

$$\textit{and} \quad \mathbb{E}\left[\mathrm{acc}_n^{\mathrm{WV}}(\mathcal{D}; \pi, \rho)\right] = \frac{1}{m} \sum_{i=1}^{m} \mathbb{I}\left[y_i = \arg\max_{|y| \leq L} \sum_{r \in \mathcal{V}^*} \pi(rey|x_i)\rho(x_i rey) \right] - \mathcal{O}(c^{-n})$$

for some constant $c > 1$.

Remarks. Theorems 1 & 2 state the convergence of the accuracy with increasing number of samples, indicating that the performance gains of using more samples will saturate for any fixed models. The limit is determined by the likelihood of generating the correct answers through all possible reasoning paths (and the likelihood should be viewed as a weighted sum for weighted majority voting). This motivates us to consider inference algorithms that search for "good" reasoning paths, such as the tree-search-based variants detailed in Sec. 3.1.1 & 3.1.2.

Theorem 1 & 2 also present insights to compare standard majority voting with weighted majority voting. Informally, as long as the reward model is "better than random", i.e., assigning higher rewards to correct solutions on average, the accuracy limit of weighted majority voting is higher than that of majority voting. In our experiments, we consistently find that weighted majority voting dominates majority voting. Thus, we focus on best-of-n and weighted majority voting in the main paper and defer majority voting results to Appendix D.

3.1.1 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 paired with a value model to score and guide the exploration steps. For additional background, we provide a review of MCTS in Appendix B.

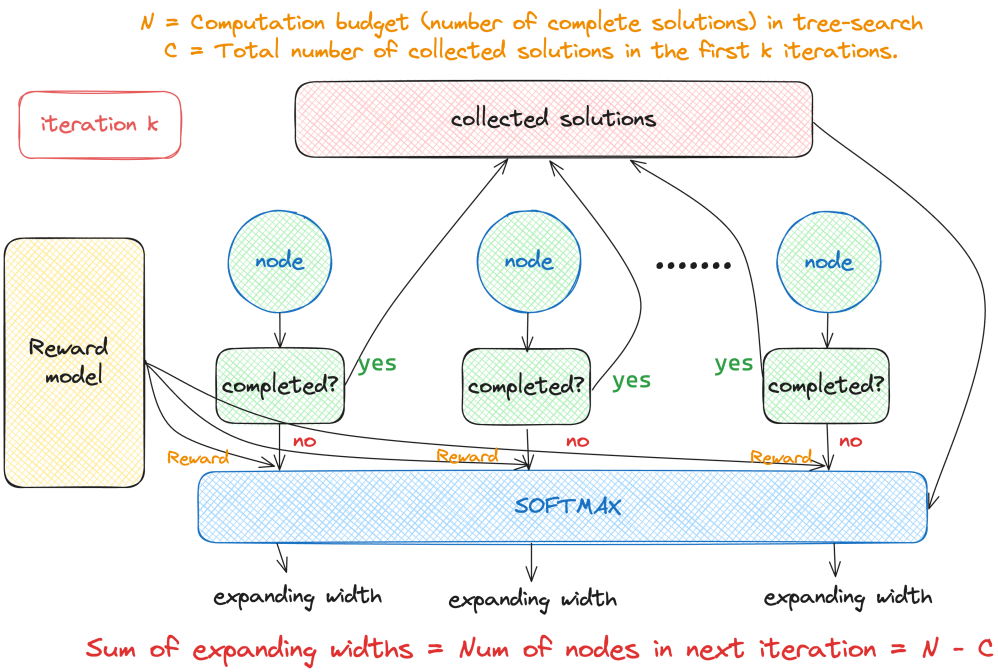

Figure 3: Illustration of one iteration of REward BAlanced SEarch (REBASE).

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 scarce or indicate potentially unfavorable cost-performance tradeoffs. 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 a new tree search method that can achieve a comparable (or better) performance as MCTS, and that is computationally less costly, with a cost similar to weighted majority voting and best-of-n. This motivates our new method, Reward Balanced SEarch (REBASE).

3.1.2 REWARD BALANCED SEARCH (REBASE)

The REBASE tree search method, illustrated in Fig. 3, inherits the exploitation and pruning properties of tree search, while using a reward model alone to estimate quality of intermediate nodes. This saves on computation compared to methods such as MCTS, since it does not involve estimate node quality with explicit rollouts. In short, the underlying idea is to use a process reward model to determine how much each node should be expanded at each depth. Namely, REBASE expands nodes at a given depth according to their softmax-normalized reward scores, subject to a total expansion budget. We describe this procedure in more detail below.

Notations. We view the fine-tuned LLM as a policy π_θ which generates the solution step by step. Given a question x and the first k steps of a solution $r_1 \cdots r_k$, the $(k+1)$-th step is sampled from $\pi_\theta(\cdot|xr_1 \cdots r_k)$. REBASE generates a solution tree during inference, in which the root node is the question x, and other nodes corresponds to solution steps. When generating solution trees, we generate children of r_k by sampling from $\pi_\theta(\cdot|xr_1 \cdots r_k)$. We use the corresponding solution step to denote a node. The reward of a node r_k is generated by the PRM: $R(r_k) := R(xr_1 \cdots r_k)$.

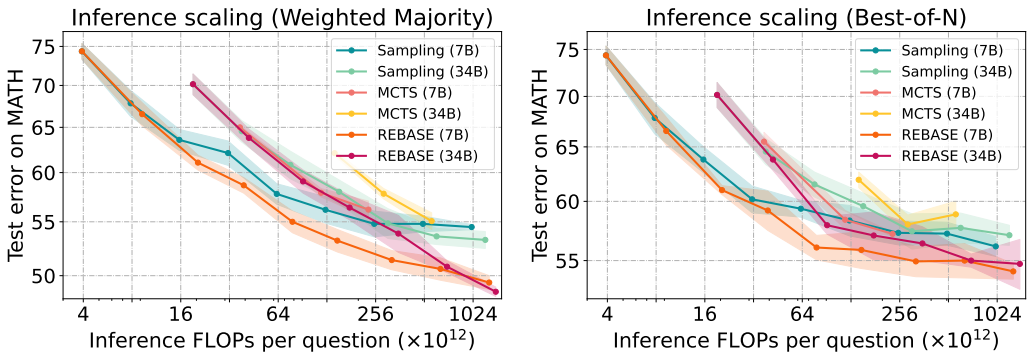

Figure 4: **MATH inference scaling across inference strategies and model sizes** (lower is better). MCTS configurations can be found in the Appendix. The left/right panel shows the error rate on MATH based on weighted majority/best-of-n. REBASE is the compute-optimal strategy at all budgets, with 7B typically the optimal model size.

Initialization. Given the question x, a balance temperature $T_b > 0$, and target number of generated 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 let the sampling budget of depth 0, B_0, to N at initialization.

Reward assignment 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:

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

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

4 Experiments

Our experiments are centered around two main questions:

- **Compute-optimal model size**: How does performance scale as inference-time compute is increased with a fixed inference strategy, but with varying model size?
- **Compute-optimal inference strategy**: How does performance scale as inference-time compute is increased with various inference strategies (and various model sizes)?

We detail our experimental setup below.

4.1 Setup

Datasets. We conduct experiments on two mathematical problem-solving datasets to investigate the effects of scaling inference compute 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.

Policy model (solution generator). To study the how performance scales as inference compute is increased using a fixed strategy, the primary axis of variation is model size. Therefore, we choose Pythia (Biderman et al., 2023) as our base models, since various model sizes are available in the Pythia family. To study inference scaling under different inference strategies (e.g., tree search, weighted majority voting), we use math-specialized Llemma models (Azerbayev et al., 2024). We finetune these models on the MetaMath dataset (Yu et al., 2024) using full parameter supervised

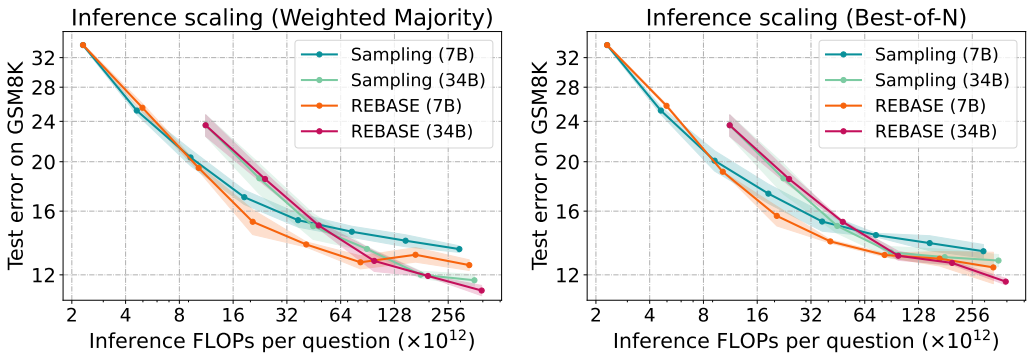

Figure 5: **GSM8k inference scaling across inference strategies and model sizes** (lower is better). The left/right panel shows the problem-solving error rate on GSM8K based on weighted majority/best-of-n. MCTS is not included in the comparison because of its poor compute-accuracy trade-off. REBASE is the compute-optimal inference strategy, and the optimal model size varies.

fine-tuning (Full-SFT), The finetuning configuration is given in the Appendix. Additionally, we test the Mistral-7B (Jiang et al., 2023) to expand our findings across different models and architectures.

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

Inference configuration. We use sampling and tree search methods to generate multiple candidates, and select the answer through best-of-n, majority voting, or weighted voting. Each configuration is run multiple times to calculate the mean and variance, which mitigates effects from randomness and thereby improves the reliability of our conclusions. Unless explicitly stated otherwise, each point in the Figures in the following section represents 2^i samples, where i is an integer starting from 0.

4.2 COMPUTE-OPTIMAL MODEL SIZE

To compare the inference compute budgets of different 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).

Scaling law of compute-optimal inference for model size. Fig. 1 shows the relationship between inference compute and error rate for different model sizes. The error rate first decreases steadily and then starts to saturate. Initially, sampling many times from smaller models is compute-optimal. At larger compute budgets the larger models are preferable, since the performance of small models has saturated. As highlighted in the right panel of Fig. 1, the optimal model size varies based on the inference budget. We performed a regression analysis on inference FLOPs C and model sizes N to establish a relationship between a given computational budget and its optimal model size. The resulting equation, $\log_{10}(C) = 1.19\log_{10}(N) + 2.03$, lets us estimate the optimal inference model size for a specific compute budget.

Llemma-7B achieves competitive accuracy to Llemma-34B with less compute. Fig. 4 and Fig. 5 shows the relationship between error rate and inference FLOPs for Llemma 7B and Llemma 34B using different inference strategies. Llemma-7B requires around $2\times$ less total FLOPs than Llemma-34B to achieve comparable accuracy. This held across inference strategies (sampling strategies, MCTS, REBASE) and tasks (MATH, GSM8K). This result suggests that, with the same training dataset and model family, generating more tokens with a suitable inference strategy using a smaller model can have more favorable cost-performance tradeoffs than using a larger model.

4.3 COMPUTE-OPTIMAL INFERENCE STRATEGY

REBASE is Pareto-optimal. REBASE consistently achieves the best cost-performance tradeoffs, outperforming the sampling-based methods in all settings when fixing the model and the evaluation task (Fig. 4, 5, 6, and 7). For example, in Fig. 4, REBASE is the compute-optimal strategy at all

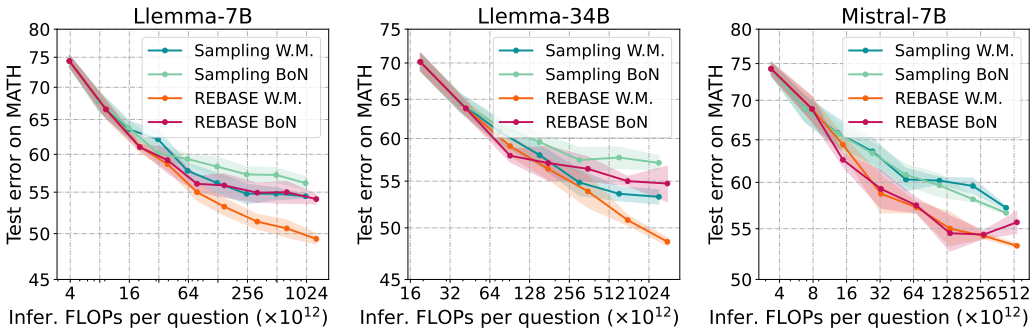

Figure 6: **MATH inference scaling across inference strategies and models** (lower is better). The tested models are Llemma-7B (left), Llemma-34B (middle), & Mistral-7B (right). In the legend, W.M. and BoN refer to weighted majority and best-of-n, respectively.

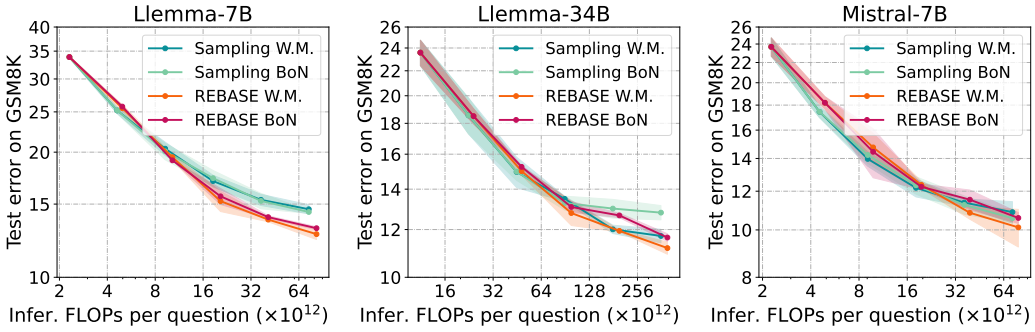

Figure 7: **GSM8K inference scaling across inference strategies and models** (lower is better). The tested models are Llemma-7B (left), Llemma-34B (middle), & Mistral-7B (right). In the legend, W.M. and BoN refer to weighted majority and best-of-n, respectively.

inference compute budgets, with 7B typically the optimal model size. On the other hand, MCTS underperforms the sampling-based methods at each compute budget, likely due to its costly rollouts (Fig. 4) compared to the efficient use of the reward model in REBASE.

Table 1 shows that REBASE achieves better accuracy with a lower compute budget compared to sampling-based weighted voting. With the 7B model, REBASE achieves higher accuracy with 7 times less compute. This finding is novel, and differs from previous tree search methods that typically improve the performance at the cost of higher computational expense compared to sampling-based voting (Chen et al., 2024a; Xie et al., 2023).

Weaker models gain more from tree search. For example, our proposed REBASE leads to 5.3%, 3.3%, and 2.6% performance gains on MATH for Mistral-7B, Llemma-7B, Llemma-34B, respectively. The order of accuracy increase is inversely related to the model's corresponding greedy search accuracy 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 accuracy. From Fig. 6 and Fig. 7, we observe that both sampling and REBASE saturate early in GSM8K and relatively late in MATH. We attribute this to the difference of in difficulty levels between GSM8K and MATH. Specifically, the LLM may assign high probability only to correct solutions in easy problems, but spread probability mass across solutions in harder problems. Thus, harder problems may require aggregating over more solution paths to converge to the distribution over answers shown in Theorems 1 & 2. On MATH (Fig. 6), we see that REBASE finally saturates with a higher accuracy than sampling. We hypothesize the reason is that drawing samples from REBASE corresponds to sampling from a policy that assigns high probability to true answers compared to sampling from the underlying language model.

Table 1: REBASE with a lower compute budget has better accuracy than sampling with a higher compute budget. We use weighted voting to aggregate candidates for both sampling and REBASE.

	# SAMPLES	FLOPS	MATH500
	MISTRAL-7B		
SAMPLING	256	8.70×10^{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**

If this was indeed the case, Theorems 1 & 2 indicate that the upper bound would become higher. We leave formally analyzing the behavior of tree search algorithms as interesting future work.

5 CONCLUSIONS, LIMITATIONS, AND DISCUSSIONS ON CONCURRENT WORKS

Conclusions. We study the relationship between task performance and the amount of compute expended during inference for various model sizes, model families, and inference strategies, to form empirical *inference scaling laws*. These relationships let us reason about *compute-optimal inference*: inference configurations that give the best performance at a given compute budget.

Our results lead to three main takeaways. First, we find that using a smaller model and generating more tokens in an inference strategy often outperforms using a larger model at a fixed compute budget. This has implications for models deployed in the real world, where inference compute is constrained in various ways. Specifically, it is potentially beneficial to deploy smaller models with more sophisticated inference strategies for better cost-performance tradeoffs. Second, we showed that in the limit of infinite compute (allocated by drawing more samples), sampling-based majority voting strategies inevitably saturate to a distribution that depends on the underlying generation policy. Hence, it is of interest to alter the sampling distribution by designing an alternative inference strategy. Third, we design such an inference strategy–the novel REBASE tree search–and find it is Pareto optimal, in that it achieves the best performance across all tested compute budgets. Notably, it outperforms commonly used weighted majority voting and MCTS methods that have attracted much interest and widespread use. This finding not only shows the strength of REBASE, but also indicates that there is large headroom to improve language model performances via inference-time algorithms.

Limitations. Our empirical analysis specifically targets mathematical problem-solving. Investigating the inference scaling laws and compute-optimal inference strategies for tasks beyond mathematical problem-solving would be a valuable direction for future research. Additionally, we mainly evaluate the proposed REBASE on the GSM8K and MATH500 datasets. We speculate that the REBASE algorithm, which assumes access only to a function that assigns scores to nodes, will be effective in tasks beyond those studied here.

REFERENCES

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

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

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.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. *arXiv preprint arXiv:2304.01373*, 2023.

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.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael

Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mo-

hammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vítor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pp. 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.

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.

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.

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.

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

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)*, pp. 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*, pp. 4195–4205, 2023.

Jacob Pfau, William Merrill, and Samuel R. Bowman. Let's think dot by dot: Hidden computation in transformer language models. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=NikbrdtYvG.

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, 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.

A Omitted Proofs

A.1 Proof of Theorem 1

Proof. Recall that we assume the answer must be shorter than L tokens. Let $\mathcal{A} = \{v \mid |v| \leq L\}$ be the set of all possible answers. Let $\tilde{\pi}(y \mid x)$ be the probability of the language model π outputting the answer y to the question x after marginalizing over the "reasoning paths", i.e.,

$$\tilde{\pi}(y \mid x) = \sum_{r \in \mathcal{V}^*} \pi(rey|x).$$

Given an input x, Assume that $y^* = \arg\max_{y \in \mathcal{A}} \tilde{\pi}(y|x)$, $y' = \arg\max_{y \in \mathcal{A} \setminus \{y^*\}} \tilde{\pi}(y|x)$, and denote

$$\delta = \tilde{\pi}(y^*|x) - \tilde{\pi}(y'|x).$$

For any y, denote by $f_n(y)$ the number of times that the model answers y in the first n samples. Let E_n be the event that majority voting with n samples does not output y^*. We note that E_n happens only if there exists y'' such that $f_n(y'') \geq f_n(y^*)$. Therefore, by union bound,

$$\begin{aligned}
\mathbb{P}(E_n) \leq & \mathbb{P}(\exists\, y'' \in \mathcal{A} \setminus \{y^*\}, f_n(y'') \geq f_n(y^*)) \\
\leq & \sum_{y'' \in \mathcal{A} \setminus \{y^*\}} \mathbb{P}(f_n(y'') \geq f_n(y^*)) \\
\leq & |\mathcal{A}| \mathbb{P}(f_n(y') \geq f_n(y^*))
\end{aligned}$$

Note that $f_n(y^*) - f_n(y')$ can be viewed as a sum of n i.i.d. random variables, which take value 1 with probability $\tilde{\pi}(y^*|x)$, -1 with probability $\tilde{\pi}(y'|x)$, and 0 otherwise. Thus, their expectations are all $\delta = \tilde{\pi}(y^*|x) - \tilde{\pi}(y'|x)$. By Hoeffding's inequality, we have

$$\mathbb{P}(f_n(y') \geq f_n(y^*)) \leq \exp\left(-\frac{n\delta^2}{2}\right).$$

Thus,

$$\mathbb{P}(E_n) \leq |\mathcal{A}| \exp\left(-\frac{n\delta^2}{2}\right) \quad \Rightarrow \quad \sum_{n=1}^{+\infty} \mathbb{P}(E_n) < +\infty.$$

By Borel–Cantelli lemma, we have

$$\mathbb{P}\left(\limsup_{n \to +\infty} E_n\right) = 0,$$

which implies the following is true *almost surely*:

$$\exists\, N \in \mathbb{N}^*, \text{ such that for any } n \geq N,\ y^* = \arg\max_{y \in \mathcal{A}} f_n(y)$$

Hence

$$\lim_{n \to +\infty} \mathrm{acc}_n^{\mathrm{MV}}(\{(x, y)\}; \pi) = \mathbb{I}\left[y = y^*\right] \qquad \text{(almost surely)}.$$

Recall the definition of y^*, the above shows the theorem is true for a dataset with a single example $\{(x, y)\}$. For general datasets $\mathcal{D}$ with m examples, one can apply the above argument to each examples and combine the results to conclude the proof of the almost-sure convergence.

Next, we prove the asymptotic result on $\mathbb{E}\left[\mathrm{acc}_n^{\mathrm{MV}}(\{\mathcal{D}\}; \pi)\right]$. We slightly abuse notation for simplicity as follows: We let $y^*(x_i) = \arg\max_{y \in \mathcal{A}} \tilde{\pi}(y|x_i)$, $y' = \arg\max_{y \in \mathcal{A} \setminus \{y^*(x_i)\}} \tilde{\pi}(y|x_i)$, and let

$$\delta_{\min} = \min_{(x_i, y_i) \in \mathcal{D}} \tilde{\pi}(y^*(x_i)|x_i) - \tilde{\pi}(y'(x_i)|x_i).$$

We denote by $E_n(x_i)$ the event that majority voting with n samples does not output $y^*(x_i)$ given input x_i. Then it's easy to see that

$$\mathbb{P}(E_n(x_i)) \leq |\mathcal{A}| \exp\left(-\frac{n\delta_{\min}^2}{2}\right) \quad \Rightarrow \quad \mathbb{P}(E_n(x_i)) = \mathcal{O}(c^{-n})$$

where $c > 1$ is a constant (which does not depend on i).

Note that if $\mathrm{acc}_n^{\mathrm{MV}}(\{x_i, y_i\}; \pi) = 1$, we have $y_i = y^*(x_i)$ unless $E_n(x_i)$ happens. In other words,

$$\begin{aligned} & \mathrm{acc}_n^{\mathrm{MV}}(\{x_i, y_i\}; \pi) \leq \mathbb{I}\left[y_i = y^*(x_i)\right] + \mathbb{I}[E_n(x_i)] \\ \Rightarrow \quad & \left|\mathbb{E}\left[\mathrm{acc}_n^{\mathrm{MV}}(\{x_i, y_i\}; \pi)\right] - \mathbb{I}\left[y_i = y^*(x_i)\right]\right| \leq \mathbb{P}(E_n(x_i)) = \mathcal{O}(c^{-n}). \end{aligned}$$

Taking a summation over the entire dataset $\mathcal{D}$ yields

$$\left|\mathrm{acc}_n^{\mathrm{MV}}(\mathcal{D}; \pi) - \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}\left[y_i = y^*(x_i)\right]\right| \leq \frac{1}{m}\sum_{i=1}^{m} \mathbb{P}(E_n(x_i)) = \mathcal{O}(c^{-n}),$$

which concludes the proof. □

A.2 Proof of Theorem 2

Proof. The proof is similar to the proof of Theorem 1. We only need to set

$$\tilde{\pi}(y \mid x) = \sum_{r \in \mathcal{V}^*} \pi(r e y | x_i) \rho(x_i r e y).$$

Then the technique in the proof of Theorem 1 immediately applies. □

B MCTS Details

In this section, we present additional background on the Monte Carlo Tree Search (MCTS) algorithm. The MCTS process can be formulated 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

$$UCT(s) = Q(s) + C\sqrt{\frac{\ln\left(N(\mathrm{Parent}(s))\right)}{N(s)}},$$

where $Q(s)$ denotes the quality score of node s, $N(s)$ is the number of visits to node s, $\mathrm{Parent}(s)$ denotes the parent node of 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:

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

C Hyper-parameters

Finetuning All the hyperparameters for model fine-tuning can be found in Table 2. We preprocess the MetaMath Dataset to make the solutions in a stepwise format.

Table 2: Fine-tuning Hyper-parameters: LR refers to the learning rate, BS refers to the batch size. Pythia, Llemma-7B and LLemma-34B are the generators we use in our experiments, RM is short for Reward Model. We only use problems from GSM8K to train the Pythia models.

Model	# Epoch	Dataset	BS	LR	Max Seq Length	Dtype
Pythia-410M	1	MetaMath (GSM8K)	128	8E-5	768	FP32
Pythia-1.4B	1	MetaMath (GSM8K)	128	4E-5	768	FP32
Pythia-2.8B	1	MetaMath (GSM8K)	128	3E-5	768	FP32
Pythia-6.9B	1	MetaMath (GSM8K)	128	2E-5	768	FP32
Pythia-12B	1	MetaMath (GSM8K)	128	1E-5	768	FP32
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

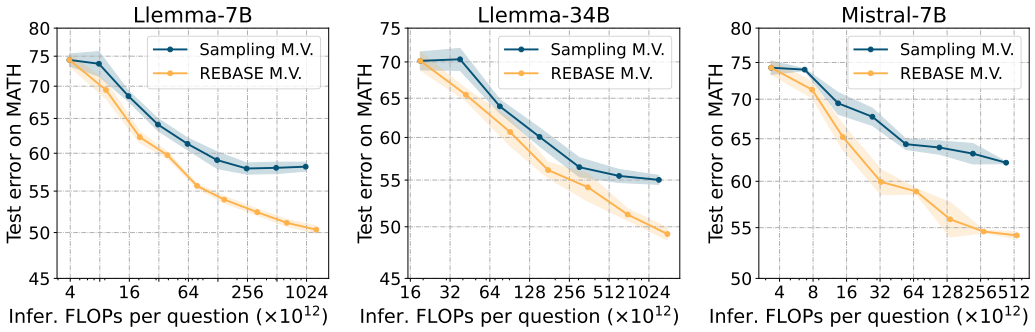

Figure 8: **The inference computation scaling laws** of different models for the problem-solving error rate on **MATH** test set. The tested models are Llemma-7B (left), Llemma-34B (middle), & Mistral-7B (right). In the legend, M.V. refer to majority voting.

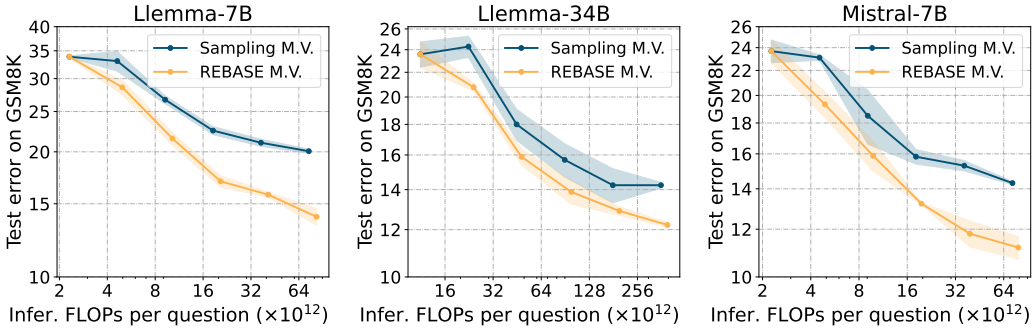

Figure 9: **The inference computation scaling laws** of different models for the problem-solving error rate on **GSM8K** test set. The tested models are Llemma-7B (left), Llemma-34B (middle), & Mistral-7B (right). In the legend, M.V. refer to majority voting.

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. New expanded nodes will be assigned values by the PRM, and then backpropagate the Q values through the process described in the last section.

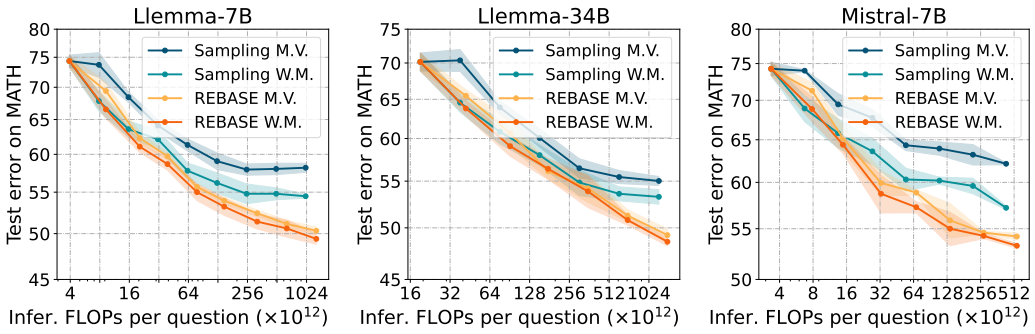

Figure 10: **The inference computation scaling laws** of different models for the problem-solving error rate on **MATH** test set. The tested models are Llemma-7B (left), Llemma-34B (middle), & Mistral-7B (right). In the legend, M.V. and W.M. refer to majority voting and weighted majority, respectively.

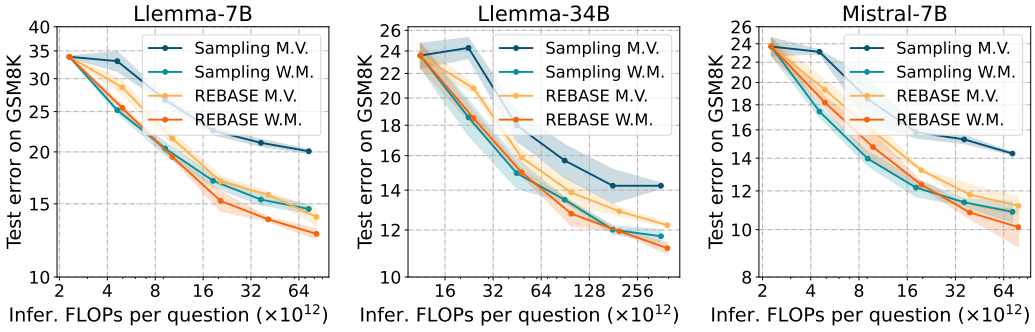

Figure 11: **The inference computation scaling laws** of different models for the problem-solving error rate on **GSM8K** test set. The tested models are Llemma-7B (left), Llemma-34B (middle), & Mistral-7B (right). In the legend, M.V. and W.M. refer to majority voting and weighted majority, respectively.

D Supplementary Figures and Tables

D.1 Majority voting experiment results

In this section, we additionally include experimental results on the majority voting method, along with its comparison with weighted majority voting (Fig. 8, 9 ,10, 11). 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.

D.2 Additional experiments on Llama3 models

We conduct additional experiments with Llama3-8B-Instruct (Dubey et al., 2024) model on MATH and GSM8K datasets, as shown in Fig. 12. Results for code generation task MBPP are presented (Austin et al., 2021) in Tab. 3. These experiments demonstrate that our conclusions generalize to the Llama3 architecture and coding tasks, confirming that increased computational effort improves performance until saturation is reached, and REBASE reaches the optimal performance-compute trade-off.

In mathematical reasoning tasks, REBASE consistently outperforms the sampling approach across different answer selection strategies, including best-of-n, majority voting, and weighted voting. The highest performance on each dataset is achieved using REBASE. Specifically, on GSM8K, REBASE combined with weighted majority voting using 128 samples achieves an accuracy of 90.2%, sur-

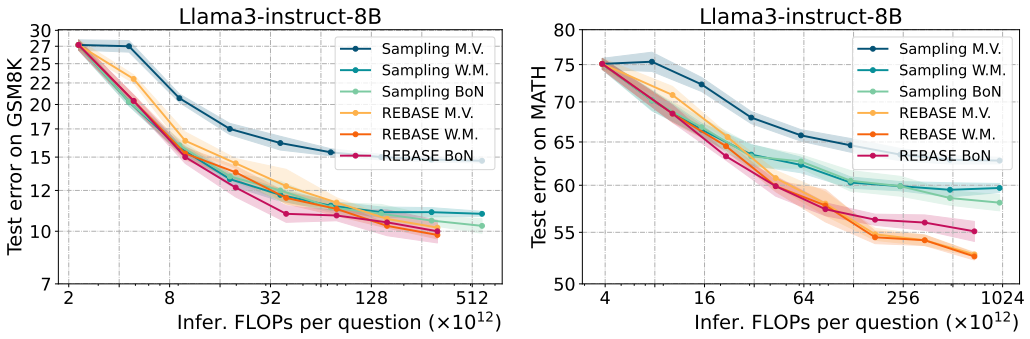

Figure 12: **GSM8K (left) and MATH (right) inference scaling across inference strategies and models** (lower is better). The tested model is Llama3-instruct-8B. In the legend, M.V., W.M., and BoN refer to majority voting, weighted majority, and best-of-n, respectively.

Table 3: Zero shot pass rates of Sampling and REBASE on MBPP code generation task.

# SAMPLES	SAMPLING FLOPS	SAMPLING PASS RATE	REBASE FLOPS	REBASE PASS RATE
8	8×10^{12}	63	8.26×10^{12}	69.6
16	16×10^{12}	69.4	17.5×10^{12}	72.4
32	32×10^{12}	72.4	34.9×10^{12}	75.8
64	64×10^{12}	79	69.15×10^{12}	81.4

passing the best accuracy of 89.7% obtained by the sampling method with 256 samples using the best-of-n strategy. Similarly, on MATH, REBASE with weighted majority voting using 128 samples achieves an accuracy of 47.4%, significantly outperforming the sampling method's best accuracy of 41.9% with 256 samples using best-of-n.

For the code generation task MBPP, we analyze scaling behavior and compute-optimal inference through pass rate evaluation. The results confirm that REBASE is more compute-efficient than sampling. This advantage can be attributed to the use of a reward model that evaluates partial code solutions. By conducting one iteration of REBASE, our method prunes suboptimal partial solutions while encouraging exploration of promising ones, thereby enhancing computational efficiency and solution quality.

D.3 COMPARISON OF DIFFERENT STRATEGIES ACROSS DIFFERENT MODELS

We show the accuracy of different stragegies under a specific compute budget in Tab. 4.

Table 4: Accuracy of different 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**
		LLAMA3-INSTRUCT-8B			
GREEDY	1	3.84×10^{12}	2.28×10^{12}	29.6	79.0
SAMPLING + MV	32	122.9×10^{12}	73.2×10^{12}	35.4	84.6
SAMPLING + BON	32	122.9×10^{12}	73.2×10^{12}	39.7	88.5
SAMPLING + WV	32	122.9×10^{12}	73.2×10^{12}	39.5	88.6
REBASE + MV	32	172.8×10^{12}	79.3×10^{12}	45.2	88.3
REBASE + BON	32	172.8×10^{12}	79.3×10^{12}	**45.5**	88.7
REBASE + WV	32	172.8×10^{12}	79.3×10^{12}	43.7	**89.1**
		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**