Do We Truly Need So Many Samples? Multi-LLM Repeated Sampling Efficiently Scale Test-Time Compute

Jianhao Chen[1 2 *] **Zishuo Xun**[2 3 *] **Bocheng Zhou**[2 *] **Han Qi**[2 *]
Qiaosheng Zhang[2] **Yang Chen**[3] **Wei Hu**[1] **Yuzhong Qu**[1 †]
Wanli Ouyang[2] **Shuyue Hu**[2 †]
[1] State Key Laboratory for Novel Software Technology, Nanjing University
[2] Shanghai Artificial Intelligence Laboratory
[3] The University of Auckland
jhchen.nju@gmail.com {whu,yzqu}@nju.edu.cn
{zhoubocheng,qihan,zhangqiaosheng,ouyangwanli,hushuyue}@pjlab.org.cn
zxun233@aucklanduni.ac.nz yang.chen@auckland.ac.nz

Abstract

This paper presents a simple, effective, and cost-efficient strategy to improve LLM performance by scaling test-time compute. Our strategy builds upon the repeated-sampling-then-voting framework, with a novel twist: incorporating multiple models, even weaker ones, to leverage their complementary strengths that potentially arise from diverse training data and paradigms. By using consistency as a signal, our strategy dynamically switches between models. Theoretical analysis highlights the efficiency and performance advantages of our strategy. Extensive experiments on six datasets demonstrate that our strategy not only outperforms self-consistency and state-of-the-art multi-agent debate approaches, but also significantly reduces inference costs. Additionally, ModelSwitch requires only a few comparable LLMs to achieve optimal performance and can be extended with verification methods, demonstrating the potential of leveraging multiple LLMs in the generation-verification paradigm.

1 Introduction

Scaling has been a major driving force of recent rapid advancements in large language models (LLMs). While scaling training-time compute [1] appears to be hitting a plateau, scaling inference-time compute stands out as a promising alternative [2]. An emerging direction is to scale inference-time compute based on the *generation-verification* paradigm. By querying an LLM with the same question for multiple times, a number of samples (or candidate answers) are generated, and then these samples are verified to deliver a final answer. Studies across various LLMs and benchmarks consistently demonstrate that simply scaling the number of generated samples significantly improves the *coverage* of correct answers [3]. Thus, it is perhaps unsurprising that recent attempts have pushed the number of samples to the scale of hundreds or even thousands [3, 4, 5], in pursuit of improved answer correctness.

However, *do we truly need so many samples*? Scaling repeated sampling is undeniably computationally expensive, with the consumption of floating point operations (FLOPs) increasing linearly with the number of samplings [6]. Additionally, in terms of user experience, repeated sampling often leads to significant delay in providing final answers [7], and no one enjoys waiting too long for a response from AI. Therefore, improving *sample efficiency* is of paramount importance, and there is a pressing need for methods that can deliver correct final answers while minimizing the number of samples required. Recent approaches have primarily focused on the verification side—a great number of outcome or process reward models [8, 9, 10] and automatic verifiers [11, 12] have been proposed, whereas LLM-as-a-judge [13, 14] has also been extensively explored.

† Corresponding author.
∗ Work done during the author's intership at Shanghai Arificial Intelligence Laboratory.
Code and data are available at `https://github.com/JianhaoChen-nju/ModelSwitch`.

Orthogonal to recent efforts, in this paper, we focus on the generation side and explore the potential of leveraging multiple LLMs to improve sample efficiency. We argue that employing multiple LLMs for generation can achieve effective complementary capabilities among the models. Trained on different corpora and using distinct paradigms, LLMs exhibit diverse capabilities—even on the same benchmark, two general-purpose LLMs may excel at answering different types of questions [15, 16]. We test our argument by building upon the simple repeated-sampling-then-voting strategy, following the *Occam's Razor* principle, and present a novel method named *ModelSwitch*. This method introduces two novel twists: (i) incorporating multiple models, even weaker ones, to produce more diverse samples, and (ii) using consistency as a signal to switch models and save compute. The rationale is based on our empirical observation: across various types of LLMs and datasets, their accuracy is positively correlated with the consistency of their generated answers. When a model generates chaotic answers, it serves as a signal to switch models. If the switched model generates consistent answers, there is a higher likelihood of obtaining the correct answer.

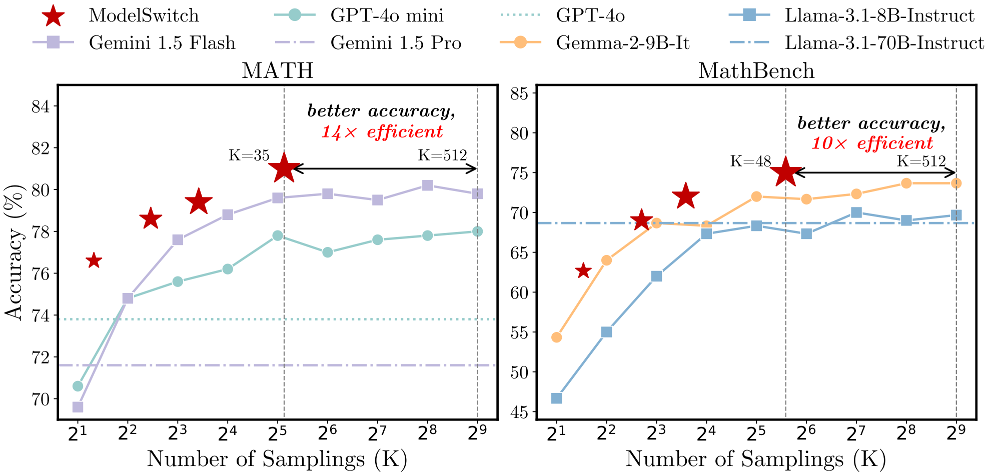

Figure 1: Performance comparison of ModelSwitch and self-consistency [17] on Math [18] and MathBench [19] dataset. ModelSwitch switches between Gemini 1.5 Flash and GPT-4o mini on MATH, and between Gemma-2-9B-It and Llama-3.1-8B-Instruct on MathBench. The curves illustrate the performance of individual LLMs under self-consistency. For comparison, horizontal lines mark the single-sample performance of larger LLMs, including GPT-4o, Gemini 1.5 Pro, and Llama-3.1-70B-Instruct, as baselines. On MATH, ModelSwitch achieves 81% accuracy with only 35 samples, outperforming Gemini 1.5 Flash (79.8% accuracy with 512 samples) while being 14× more efficient. On MathBench, similar results are observed with open-source models: ModelSwitch achieves 75% accuracy (48 samples), outperforming Gemma-2-9B-It (73.7%, 512 samples) with 10× efficiency. Additionally, combining 9B and 8B models achieves performance (69%) comparable to a 70B model (68.7%) with only 7 samples.

We conduct extensive evaluations on six datasets that cover a wide range of knowledge, reasoning, comprehension, and domain-specific challenges. ModelSwitch, when leveraging Gemini 1.5 Flash and GPT-4o mini, significantly surpasses the self-consistency [17] performance of each respective LLM. Moreover, it even outperforms more advanced LLMs such as GPT-4o and Gemini 1.5 Pro—achievements that self-consistency cannot match. At the same time, ModelSwitch demonstrates superior sample efficiency, reducing the average sampling count per query by 34% on six datasets, showcasing its ability to save computational costs while maintaining high performance. Furthermore, ModelSwitch achieves state-of-the-art performance on five datasets, outperforming five other multi-agent or multi-LLM methods, and secures the second-best result on the remaining dataset. Notably, ModelSwitch achieves an impressive 63.2% accuracy on the MMLU-Pro [20] dataset, surpassing the best single LLM by 10.2 points and significantly exceeding other methods, including MAD [21] (45%), AgentVerse [22] (38%), ChatEval [23] (43%), MAD (multi-LLM version) (50.2%), and MOA [16] (50.8%). In additional experiments, we demonstrate that ModelSwitch requires only a small number of LLMs with comparable performance to achieve optimal results and is relatively robust to the order of models. We also show that ModelSwitch can be integrated with verification methods, such as reward models, further enhancing the performance.

We conduct a formal analysis to better understand the improvements brought by ModelSwitch, compared to self-consistency. For a given query, we derive a sufficient condition and a necessary condition that ModelSwitch (two models) can surpass the performance of single model. This sufficient condition may still be satisfied even when neither of the two models can produce a completely consistent answer lists. This reveals the source of the superiority of ModelSwitch compared to a single model. Furthermore, we provide a theoretical bound on the efficiency gains achieved by ModelSwitch, offering deeper insights into its advantages. In particular, if the probability of each model producing a completely consistent answer list is greater than some constant $c > 0$ and each model is sampled the same number of times, the expected number of samplings can be decreased by a factor $\frac{1}{n}\frac{1-c}{c}$, where n is the number of models.

In summary, our key contributions are outlined as follows:

1. An empirical analysis that reveals a universal correlation between consistency and accuracy of generated answers across various popular LLMs and datasets.
2. A simple, effective, and cost-efficient generation-verification method that effectively leverages the complementary strength of multiple LLMs.
3. Extensive experiments demonstrating that ModelSwitch outperforms single-LLM sampling-then-voting in efficacy and efficiency, and more effectively leverages multi-LLM synergies compared to debate-based approaches.
4. A theoretical analysis that provides a deeper understanding of why multiple-LLM generation is superior to single-LLM generation.

2 A Universal Correlation between Consistency and Accuracy

In this section, we conduct an empirical analysis of the relationship between consistency (in terms of the entropy of the generated answers) and accuracy of multiple popular LLMs. Self-consistency [17] has observed that the consistency (here measured as the percentage of decodes agreeing with the final aggregated answer) is highly correlated with accuracy on the GSM8K dataset. However, this observation was limited to a single dataset and a single model. To verify the generality of this finding, we extend our analysis to multiple datasets and several mainstream LLMs. Specifically, we use entropy to quantify the consistency among different generated answers. Compared to the percentage of decodes agreeing with the final aggregated answer, entropy provides a more comprehensive representation as it provides a more accurate characterization of the generated answer distribution.

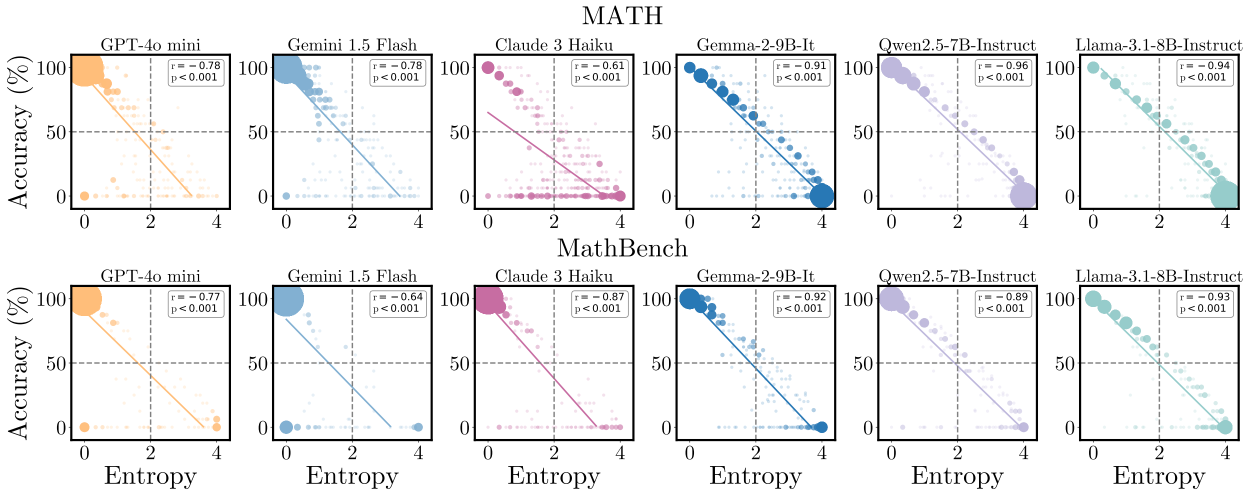

Figure 2: Correlation between consistency (entropy) and accuracy of answers from six LLMs on MATH and MathBench. We use the transparency and size of the scatter points to indicate the number of queries corresponding to each point (the larger and more opaque the point, the greater the quantity). We report the correlation coefficient r and the significance indicator p. In each subplot, entropy and accuracy exhibit a moderate $(0.5 < |r| \leq 0.8)$ or high $(|r| > 0.8)$ correlation, and this correlation is statistically significant $(p < 0.001)$.

As shown in Figure 2, we fix each LLM to sample 16 times per query, and calculate the entropy and the accuracy of the 16 answers. Consistency (entropy) and accuracy exhibit a strong correlation (r), and this trend is universal across all six LLMs, irrespective of their different sizes and performances. There are three extreme cases that can be discussed separately: those distributed in the top-left (high consistency, high accuracy), bottom-left (high consistency, low accuracy), and bottom-right (low consistency, low accuracy). Distribution in the bottom-right indicates that the model generated 16 entirely different answers, suggesting a lack of confidence in the current query. In such cases, even if a correct answer exists, it is difficult to be selected without oracle verifiers. Distribution in the top-left and bottom-left means the model generated only one type of answer, indicating high confidence. However, the difference is that the former is confidently correct, while the latter is confidently wrong. The points in the top-left greatly outnumber those in the bottom-left, which means that when an LLM generates highly consistent answers for a query, there is a strong likelihood that the answer is correct. *In a word, consistent implies correct, while chaotic implies wrong*.

3 ModelSwitch: Harnessing Consistency in Multi-LLM Generation

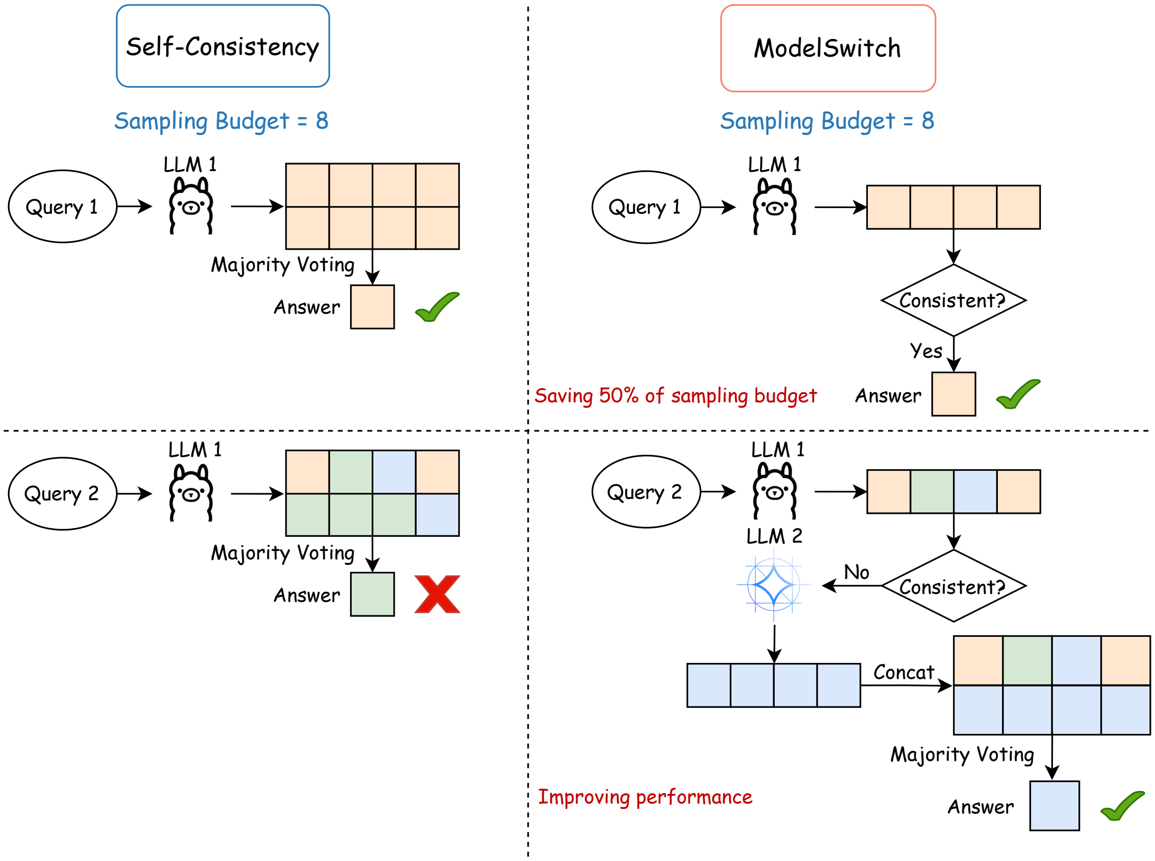

Figure 3: An overview of how ModelSwitch approach works between two LLMs. Given a sample budget K, it first queries the first LLM with $\frac{K}{2}$ samples. If self-consistency is achieved, the answer is accepted, saving 50% of the budget. Otherwise, it queries the second LLM with left $\frac{K}{2}$ samples and aggregates answers from both models, potentially improving performance if the second LLM excels at the task.

ModelSwitch is a simple yet effective framework for multi-LLM repeated sampling that simultaneously achieves high performance and efficiency. Based on the analysis in Section 2, our core idea is to leverage the strong correlation between consistency and accuracy during repeated sampling. When consistency is high, we can reduce the number of samplings and confidently rely on the current answer. Conversely, when consistency is low, it indicates that the model "does not know" the answer. In such cases, rather than persisting with the same model, we can switch to another model, embracing the possibility that "the next model you encounter might know what the previous one did not." This approach enables complementary strengths across models, optimizing both efficiency and accuracy. Combining these insights, we present our ModelSwitch design, detailed in Algorithm 1.

ModelSwitch replaces a single LLM with multiple LLMs during the sampling stage. First, the selection of LLMs plays a crucial role in the effectiveness of this approach. We prioritize diverse models over different versions of the same one. This is because similar versions often share overlapping knowledge, which limits diversity and reduces the potential for complementary insights. The next step in this approach is determining how to allocate the sampling budget across the models. In our method, we simply set the sampling budget for each LLM to be equal. This means that, given a

Algorithm 1 ModelSwitch Algorithm

```
Input: Model set $\{M_1, M_2, \ldots, M_n\}$, sampling budget = $K$
Output: $Answer$
Initialize $All\_M.answers = \{\}$
for $i = 1$ to $n$ do
    Initialize $M_i.answers = \{\}$
    for $j = 1$ to $\frac{K}{n}$ do
        $sample = M_i$.sampling()
        $answer$ = Extract_Answer($sample$)
        Add $answer$ to $M_i.answers$
    end for
    if $i \leq n - 1$ and size(set($M_i.answers$)) $== 1$ then
        $Answer = M_i.answers(0)$ and Exit
    end if
    Add $M_i.answers$ to $All\_M.answers$
end for
$Answer =$ voting_algorithm($all\_results$)
```

fixed total sampling budget, instead of a single model sampling K times, we distribute the budget evenly across n models, allowing each to sample $\frac{K}{n}$ times without increasing the total sampling budget. Empirically, when performing ModelSwitch, it is best to arrange the models from strongest to weakest overall. If a model earlier in the order can answer the current query, the remaining LLM calls are saved, which improves computational efficiency.

Weighted Voting Algorithm Following the sampling-then-voting strategy, the most common method is majority voting, where the most frequently occurring answer among all LLMs is selected as the final answer. However, when a large quantity of models are involved in voting, performance differences often exist among them, so equally aggregating the answers from all models is not an optimal approach. Therefore, we design a weighted voting algorithm with internal weights W_α and external weights W_β. Each answer will be multiplied by these two weights when voting.

Internal weights W_α measure a model's confidence in its own answers. If a model consistently produces the same answer across generated samples, its internal weight is higher. Note that W_α is automatically calculated instead of a hyperparameter. We use entropy to calculate the consistency and normalize the internal weights to ensure that they fall within the same range for each model. As shown in Formula 1, we scale W_α to $[\text{bias}, 1]$ by using normalization and bias, where norm = $\log_2$ len(answer set), bias = $\frac{1}{\text{len(samples)}}$. In the worst case, the answer set equals the samples size; in the best, it reduces to one.

$$\mathrm{W}_\alpha = \text{bias} + (1 - \text{bias})\left(1 - \frac{\text{Entropy}}{\text{norm}}\right), \text{ if norm} > 0 \text{ else } 1 \tag{1}$$

Consider that we have 3 model answer sets denoted as M_1, M_2, M_3. $\mathrm{M}_1 = \{\mathrm{A} * 3, \mathrm{B} * 2\}$, $\mathrm{M}_2 = \{\mathrm{B} * 3, \mathrm{A} * 2\}$, $\mathrm{M}_3 = \{\mathrm{C} * 5\}$. According to the majority voting algorithm, answers A, B, and C have the same score. However, since M_3 has higher confidence, the score of C becomes higher after incorporating internal weights W_α, allowing it to stand out. Our goal is to ensure that the answers of the more confident models receive higher scores.

External weights W_β assess prior performance, ensuring models with significantly different performances have corresponding importance in voting. For example, we can assign $\mathrm{W}_\beta = 2$ or 1.5 to much stronger models, while giving $\mathrm{W}_\beta = 1$ to weaker models. Specifically, when there are only two models, we set $\mathrm{W}_\beta =$ uniformly to 1.

Special Case Two-LLM switch is a very special case, as shown in Figure 3, not only because it is easy to set up but also because it has good properties under specific conditions. When the sampling counts for the two models are equal (denoted as $\frac{K}{2}$), the effect of ModelSwitch under the sampling budget K is exactly equivalent to each model sampling $\frac{K}{2}$ completely and then mixing. This means that we achieve completely lossless performance while greatly enhancing efficiency. This property is straightforward to prove: consider two models, M_1 and M_2, each generating $\frac{K}{2}$ samples. Let A_1 be the most frequent answer from M_1, with frequency $f(A_1)$. The performance remains lossless because:

- If $f(A_1) < \frac{K}{2}$, the samples of both models are considered.
- If $f(A_1) = \frac{K}{2}$, A_1 is selected, and M_2's samples are pruned without affecting the final answer. Assuming a tie-breaking rule that favors M_1's answer.

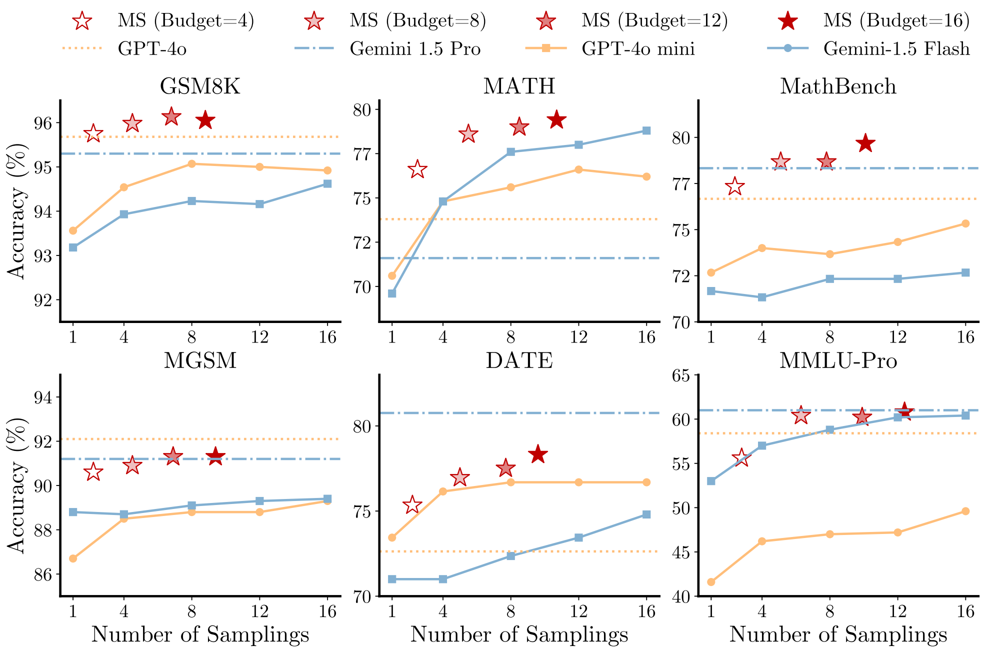

Figure 4: Performance comparison of self-consistency for each LLM (GPT-4o mini and Gemini 1.5 Flash) and ModelSwitch using both. We use horizontal lines to mark the single-sample results of more advanced LLMs GPT-4o and Gemini 1.5 Pro. We use the shade and size of red stars to differentiate the sampling budgets of ModelSwitch. The horizontal coordinate of the red star reflects the actual sampling counts of ModelSwitch.

4 Experiments: On the Efficacy, Efficiency, Scalability, and Robustness of ModelSwitch

Our experiments aim to answer the following four questions: (i) Does ModelSwitch outperform single-LLM repeated-sampling in terms of efficacy and efficiency? (ii) Can ModelSwitch surpass other multi-agent (LLM) debate methods? (iii) Does scaling the number of LLMs involved in ModelSwitch lead to continued performance improvements? (iv)Can ModelSwitch be extended with stronger verification methods?

4.1 Experimental Setup

Benchmarks We mainly evaluate our method on the following six datasets that cover a wide range of knowledge, reasoning, comprehension, and domain-specific challenges, providing a robust evaluation for our method: GSM8K [8], MATH [18], MathBench [19], MGSM [24], DATE [25], and MMLU-Pro [20]. Details of the six datasets are provided in Appendix A.1.

Models We primarily consider three lightweight, closed-source LLMs: GPT-4o mini [26], Gemini 1.5 Flash [27], and Claude 3 Haiku [28]. In Section 4.2, we compare ModelSwitch to single-LLM repeated sampling methods, using GPT-4o mini and Gemini 1.5 Flash, while also including more advanced LLMs, GPT-4o and Gemini 1.5 Pro, as additional baselines. Subsequently, in Section 4.3, we use all three lightweight LLMs to evaluate ModelSwitch against other multi-agent debate methods. In Section 4.4, we investigate the impact of the number of LLMs by combining the three aforementioned lightweight closed-source LLMs with three open-source LLMs—Llama-3.1-8B-Instruct [29], Gemma-2-9B-It [30], and Qwen2.5-7B-Instruct [9]—to assess how performance scales with model diversity. Finally, in Section 4.5, we explore whether ModelSwitch can be enhanced by integrating stronger validation methods, using Qwen2.5-MATH-RM-72B. For details on all hyperparameter settings, see Appendix A.2.

4.2 ModelSwitch Outperforms Single-LLM Sampling-then-Voting in Efficacy, Efficiency and Scalability

In this section, we compare ModelSwitch with self-consistency, which is the classic approach for single-LLM sampling-then-voting. To ensure fair comparison, we evaluate self-consistency for each LLM (GPT-4o mini and Gemini 1.5

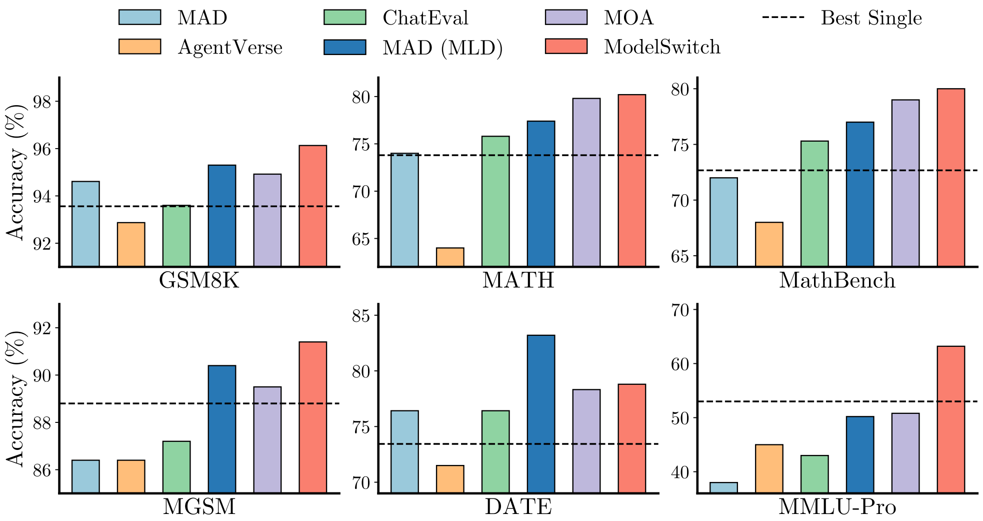

Figure 5: Performance comparison of different multi-agent debate systems under the same budget of 15 samples relative to the accuracy of the best single LLM with one sample. ModelSwitch achieves the best results on five datasets and the second-best result on DATE.

Flash) and ModelSwitch using both, under the same sampling budget. All our queries are asked in COT [25] format by default. The results are shown in Figure 4.

Efficacy Across all the benchmarks considered, ModelSwitch with two LLMs outperforms single-LLM self-consistency, even when using the best-performing single LLM. For instance, as the sampling budget increases from 1 to 16, ModelSwitch delivers a 7-point performance boost on MathBench (increasing from 72.7% to 79.7%), significantly outperforming the self-consistency gains of Gemini 1.5 Flash at 2.6-point (72.7% to 75.3%) and GPT-4o mini at 1-point (71.7% to 72.7%). On MMLU-Pro, even when switching between models with a performance gap exceeding 10%, we can still achieve an improvement over self-consistency of the best-performing single LLM.

Efficiency While ModelSwitch outperforms self-consistency, it also requires fewer samples. Taking the sampling budget of 16 as an example, ModelSwitch reduces the average actual sampling count per query to 9.2, 10.7, 10.5, 9.4, 10.6, and 13.4 across the six datasets. (Note that since we calculate the average sampling counts per query across the entire dataset, the actual sampling counts for ModelSwitch may not necessarily be integers.) *Overall, ModelSwitch saves 34% samples on average, and saves up to 43% on GSM8K and 41% on MGSM, respectively.*

Scalability ModelSwitch leveraging GPT-4o mini and Gemini 1.5 Flash, exhibits superior scalability compared to self-consistency for each LLM, outperforming more advanced LLMs GPT-4o and Gemini 1.5 Pro. For example, with an average of 2.2 samples per query on GSM8K, ModelSwitch surpasses both GPT-4o and Gemini 1.5 Pro. *Similarly, on MathBench, ModelSwitch outperforms both models with an average of 5.1 samples per query—achievements that self-consistency fails to match.*

4.3 ModelSwitch Better Leverages Multi-LLM or Multi-Agent Synergies than Debate Methods

In this section, we evaluate our approach against other multi-agent debate methods that have demonstrated competitive performance: MAD [21], selected as a canonical representative of the multi-agent debate paradigm, establishes a strong baseline for debate-style collaboration through its innovative use of inter-agent critique to enhance factuality and reasoning capabilities; ChatEval [23], chosen for its extension of the MAD framework, showcases how multi-agent collaboration can effectively mimic human evaluation processes, particularly in complex assessment tasks; AgentVerse [22], which emphasizes dynamic group composition and the emergence of social behaviors among agents, enabling more adaptive and synergistic collaboration in diverse scenarios; and MOA [16], which includes a novel hierarchical architecture that aggregates samples from multiple LLM agents engaging in critical thinking through mutual evaluation, and which we perceive as a form of implicit debate. For MAD and AgentVerse, we use a single LLM, GPT-4o mini. We additionally construct a new baseline, referred to as MAD (MLD), by extending MAD into a multi-LLM debate version. In ChatEval and MAD (MLD), we conduct debates using GPT-4o mini and Gemini 1.5

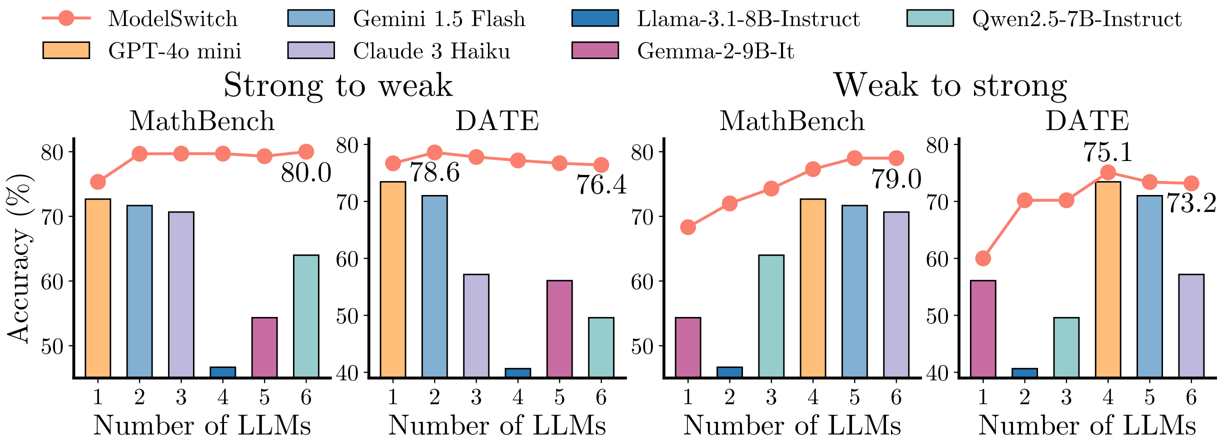

Figure 6: Performance of scaling the number of LLMs under two settings: strong-to-weak and weak-to-strong. The bars show the performance of each individual model with one sample, whereas the curves show the performance of ModelSwitch when involving the models represented by the current and all previous bars. All data in the figure are results obtained under the same sampling budget of 16 samples. In each subplot, we have marked the optimal performance and the performance where all models are included. Few, competing LLMs are sufficient to achieve the optimal performance of ModelSwitch. Moreover, ModelSwitch is relatively robust to order when containing same models.

Flash. In MOA and ModelSwitch, we utilize GPT-4o mini, Gemini 1.5 Flash, and Claude 3 Haiku. Due to the varying architectures of different systems, a relatively fair comparison is to unify the sampling budget across all systems when evaluating their performance. We set the total number of LLM sampling budget for all systems to 15. For specific settings of different systems, see Appendix A.3.

Efficacy As shown in Figure 5, our method achieves state-of-the-art performance on five datasets, with a significant lead on MMLU-Pro. For example, MAD (45%), AgentVerse (38%), ChatEval (43%), MLD (50.2%) and MOA (50.8%) all perform worse than the best single LLM (53%) sampling once on MMLU-Pro. This performance gap highlights a key limitation of these systems: while they rely on interactions between multiple agents (LLMs), it is difficult to ensure that such interactions consistently guide the answers in the correct direction. Challenges like error propagation often arise during these interactions [15, 31], which are especially hard to mitigate on a complex dataset like MMLU-Pro. In contrast, ModelSwitch achieves a remarkable 63.2% accuracy, representing an impressive 10.2-point increase over the best single LLM. This demonstrates its ability to effectively address these issues and deliver superior performance. This also highlights the potential of exploring how to effectively promote collaboration among different LLMs as a promising research direction.

4.4 ModelSwitch Needs Only Few, Comparable LLMs and Remains Robust to LLM Order

In this section, we investigate the impact of the number and order of LLMs on the performance of ModelSwitch. As shown in Figure 6, we experimented with scaling the number of LLMs involved from 1 to 6. To study the effect of order on performance when increasing the number of models, we used two settings: strong-to-weak and weak-to-strong. During this scaling process, we consistently maintained the sampling budget at 16, aiming to demonstrate the potential of expanding along the dimension of model quantity under the same sampling budget. We selected three closed-source LLMs with relatively close performance and three open-source LLMs also with relatively close performance. There is a significant performance gap between the closed-source and open-source LLMs. It would be interesting to observe whether there is a sharp decline in performance when strong models are combined with weaker ones.

Effects of LLM Number Under the strong-to-weak setting, the most significant improvement in performance is observed when scaling from 1 model to 2 models. However, from 2 to 6 models, the performance either plateaued (79.7%$->$ 80.0% on MathBench) or declined (78.6%$->$ 76.4% on DATE). This indicates that simply increasing the number of models does not continuously enhance performance. Selecting few, comparable LLMs for ModelSwitch yields the best results.

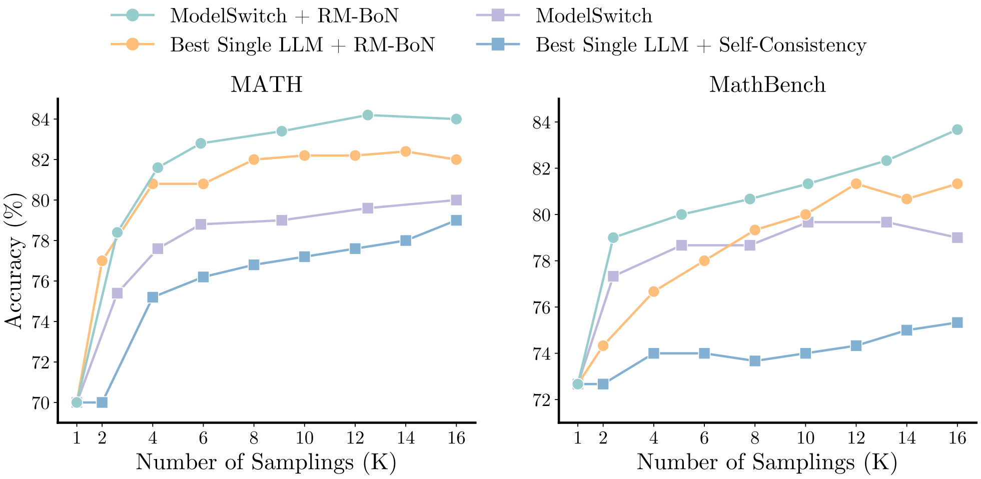

Figure 7: Performance comparison of ModelSwitch and single LLM combined with Qwen2.5-MATH-RM-72B as the reward model using the Best of N strategy (denoted as RM-BoN). We use "Best Single LLM" to refer to the best-performing model on each dataset: specifically, Gemini 1.5 Flash on MATH and GPT-4o mini on MathBench. ModelSwitch, integrating GPT-4o mini and Gemini 1.5 Flash, achieves superior accuracy on both MATH and MathBench datasets, outperforming the best single LLM under the same RM-BoN strategy.

Effects of LLM Order Comparing the strong-to-weak setting and the weak-to-strong setting, the strong-to-weak setting achieves better overall performance unsurprisingly. Notably, on two datasets, the performance of the weak-to-strong arrangement does not decrease significantly ($80.0\% -> 79.0\%$, $76.4\% -> 73.2\%$). The results from the weak-to-strong arrangement also suggests the relative robustness of our method to the order of models.

4.5 ModelSwitch Can be Combined with Stronger Verification For Performance Boost

In this section, we combine ModelSwitch with Qwen2.5-MATH-RM-72B [9], a highly capable mathematical reward model, to score different samples from GPT-4o mini and Gemini 1.5 Flash. We utilize the Best of N (BoN) strategy to select the answer with the highest score instead of voting . Specifically, our approach is to replace the voting_algorithm (all_results) in Algorithm 1 with BoN (all_results). We compare the performance curves of the best single LLM and ModelSwitch under RM-BoN strategy. The results are shown in Figure 7.

ModelSwitch, when combined with stronger verification method (RM-BoN), achieves a notable improvement in performance (with 16 samplings, accuracy increases from 80% to 84% on MATH and from 79% to 84% on MathBench). At the same time, it consistently outperforms the best single LLM repeated sampling + RM-BoN (with 16 samplings, ModelSwitch + RM-BoN achieves 84% accuracy on MATH vs. 82% for the best single LLM + RM-BoN, and 84% on MathBench vs. 81.33%). Interestingly, on MathBench, vanilla ModelSwitch even surpasses the best single LLM + RM-BoN given a small number of samplings. Specifically, with a sampling count of 5, ModelSwitch achieves a 78.7% accuracy, outperforming the best single LLM + RM-BoN with 6 samplings (78%), demonstrating the efficiency advantages of our method.

Overall, the ModelSwitch approach primarily emphasizes strategies in the generation (sampling) phase, thus it can be further enhanced by stronger selection (verification) methods to improve overall performance. Multi-LLM generation still holds significant potential.

5 Understanding Why ModelSwitch Outperforms Self-Consistency

In this section, we analyze when ModelSwitch can improve performance for a given fixed query. For the analysis on the entire dataset, please refer to the Appendix C. To simplify the problem, we only analyze the case of two models M_1 and M_2. We assume that the first option is the correct answer. Let $p_1 = (x_1, ..., y_1, ...)$ and $p_2 = (x_2, ..., y_2, ...)$ denote the output distributions of M_1 and M_2 for some fixed query, where x_1 and y_1 are defined as follows (the definition of x_2 and y_2 are similar): If x_1 is the largest one in the probability vector p_1, then y_1 is the second largest; otherwise, y_1 is the largest.

Assume that the sampling counts for the two models are equal. We focus on the expected performance, we can directly compare two probability vectors to find out the performance difference between the two mixed models and a single model. The probability of correct answer of ModelSwitch can be denoted as $P = x_1 + x_2$. If P is the largest one in the probability vector $p_1 + p_2$, the ModelSwitch can obtain the correct answer (in the sense of expectation). For situations where both M_1 and M_2 provide the correct final answer or both give the same incorrect final answer, ModelSwitch does not change the final answer so we do not consider these two situations. Let $Q = y_1 - y_2$ and the number of options is denoted as m. We have the following proposition.

Proposition 5.1. *The necessary condition for ModelSwitch to obtain the correct answer is $P > \frac{2}{m}$. The sufficient condition is*

$$P + Q + x_1 > 1 \textit{ and } P - Q + x_2 > 1.$$

Proof. (1) *Sufficiency*: From the definition of x_1 and y_1 (x_2 and y_2), y_1 and y_2 is the largest or second largest in p_1 and p_2. To ensure P is the largest one in the probability vector $p_1 + p_2$, P only needs to be greater than the sum corresponding to y_1 and y_2. Thus we have, $P > y_1 + 1 - x_2 - y_2$ and $P > y_2 + 1 - x_1 - y_1$.

(2) *Necessity*: If $P \leq \frac{2}{m}$, the average value of the remaining part of $p_1 + p_2$ after removing $x_1 + x_2$ is greater than $\frac{2-2/m}{m-1} = \frac{2}{m}$. There exists some value in $p_1 + p_2$ that is greater than $\frac{2}{m}$. Hence, ModelSwitch cannot get the correct answer when $P \leq \frac{2}{m}$.

□

The sufficient condition in Proposition 5.1 can be intuitively explained as follows: P should be as large as possible, which is the direct factor that enables ModelSwitch to yield the correct answer. On the other hand, $|Q|$ should not be overly large, as this would result in at least one of the inequalities in the sufficient condition failing to hold. This is because if $|Q|$ is too large, it implies that the probability of an incorrect answer from a particular model (M_1 or M_2) has an overwhelming advantage, making it difficult to enhance performance through mixing.

Note that, this proposition does not impose constraints on whether M_1 and M_2 can obtain the correct final aggregated answer individually, i.e., it does not require x_1 or x_2 to be the largest probabilities. Even both models cannot obtain the correct final aggregated answer, the above proposition indicates that ModelSwitch still have chance to obtain the correct final aggregated answer. This is why repeated sampling with multiple LLMs yield better results than with a single LLM.

For example, if $p_1 = (0.4, 0.6, 0)$ and $p_2 = (0.4, 0.0.6)$, both M_1 and M_2 cannot obtain the correct answer, while the mixed model can. This example illustrates that ModelSwitch can reduce the system's confidence in each incorrect answer. Thus, **by ModelSwitch, different errors in various models can counteract each other**. For another example, if $p_1 = (0.7, 0.1, 0.2, 0)$ and $p_2 = (0.15, 0.05, 0.05, 0.75)$, M_1 can obtain the correct answer but M_2 can't. Then the mixed model can output the correct answer. It can be verified that both examples satisfy Proposition 5.1.

Computational Efficiency Consider a set of n models, denoted as $\{M_1, M_2, \ldots, M_n\}$. For a given query and a total sampling budget K, each model is sampled sequentially up to $\frac{K}{n}$ times, such that the theoretical maximum number of samples is K. Each model M_i has a probability P_i of producing a completely consistent answer list. If a model produces such a consistent answer list, subsequent models are pruned, and no further sampling is performed.

In this scenario, the **expected actual number of samplings** ($\mathbb{E}[N_{\text{call}}]$) can be bounded as follows:

$$\mathbb{E}[N_{\text{call}}] = \frac{K}{n} \sum_{k=1}^{n} \prod_{i=1}^{k} (1 - P_i). \tag{2}$$

If the probability of each model producing a completely consistent answer list is greater than some constant $c > 0$, the expected number of samplings can be bounded by $\frac{K}{n} \frac{1-c}{c}$, which reduces the number of samplings by a factor $\frac{1-c}{nc}$.

The **expected cost savings**, representing the proportion of computation cost saved through ModelSwitch, can then be expressed as:

$$\text{Expected Cost Savings} = \frac{K - \mathbb{E}[N_{\text{call}}]}{K}. \tag{3}$$

6 Related Work

Two lines of work are most relevant to ours: generation-verification paradigm and multi-agent collaboration. Generation-verification paradigm follows a two-stage pattern: generating multiple candidate answers through sampling [3, 32], then selecting the most suitable ones using verification mechanisms [8, 11]. Multi-agent collaboration [33, 34, 35, 36, 37] enhances answer quality through collaborative interactions among multiple reasoning agents powered by single or multiple LLMs.

Generation-Verification Paradigm. The generation-verification paradigm underlies most recent advances of test-time compute. Self-consistency [17] involves sampling various reasoning paths and selecting the most consistent answer, thereby enhancing chain-of-thought reasoning. It is arguably the simplest generation-verification strategy, and has been shown to be a highly effective test-time compute strategy in LLMs. For instance, GPT-o1 [38] and Deepseek-R1 [39] have shown notable improvements when using self-consistency as a test-time compute strategy. This method Recent advances enhance this paradigm through structured decomposition: Universal Self-Consistency [40] directly generates final answers from N-best candidates via LM prompting, while Branch-Solve-Merge [41] decomposes problems into sub-prompts and merges solutions through meta-prompting. Similarly, Li et al. [42] emphasize that increasing the number of samplings can outperform larger LLMs. Brown et al. [3] show that increasing the number of samplings consistently improves the coverage of correct answers. Our work aims to enhance generation effectiveness and efficiency by using multiple LLMs to complement each other.

Beyond basic methods like voting, verification strategies have evolved significantly: *Reward-based selection* [43, 44, 45, 46, 10, 47, 5] significantly improves answer selection by learning to rank generated samples based on quality; *Automatic verification* leverages tool calls (e.g., calculators, search engines, code executors) [11, 12] or mathematical proof-checking [48] for rigorous validation; *LLM-as-a-Judge* [13, 49, 14] improves verification accuracy through off-the-shelf LLMs which aligns closely with human preferences. We employ a default voting strategy that balances simplicity and effectiveness, and it has been shown to be enhanced by strong verification methods.

Multi-Agent Collaboration. A recently emerging class of multi-model collaboration methods is multi-agent debate. Multi-agent debate [21, 23, 22, 50, 51, 52, 53] assigns multiple agents of the same LLM as proponents and opponents, engaging them in role-playing dialogues to refine answers through iterative debate. However, these debate methods typically rely on single homogeneous LLM. Recently, mixture of agents (MOA) [16, 54] refines samples by critically evaluating answers across different LLMs in iterative cycles. The commonality of multi-agent debate systems is that different agents (LLMs) are forced to influence each other. While showing promise, under what circumstances LLMs agree on the correct answer and when they are swayed by noise remains an open question without a well-established conclusion.

Another class of multi-agent collaboration approach is model routing [55, 56, 57, 58, 59], which trains router networks to distribute questions to specialized models. Our method achieves analogous benefits through dynamic model switching based on consistency – effectively implementing a *training-free router* that adaptively selects the most appropriate model per query.

Our work combines generation-verification paradigm and multi-agent collaboration by proposing a framework that leverages model diversity without relying on explicit agent interactions.

7 Conclusion

In this paper, we leverage the complementary strengths of multiple LLMs at the test time without requiring internal fusion or additional training to improve performance. Building on the repeated-sampling-then-voting framework, we introduce a strategy that not only enhances performance but also significantly improves computational efficiency.

Empirical observations demonstrate a strong correlation between a model's internal consistency and the accuracy of its generated final answers, laying the foundation of our approach. Extensive experiments confirm that our method ModelSwitch achieves competitive performance while substantially reducing inference costs. Theoretical analysis further validates the efficiency and performance benefits. This makes our strategy a practical and generalizable solution for various reasoning and knowledge-based tasks, paving the way for more efficient and effective applications of LLMs.

References

[1] Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

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

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

[4] Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. Are more llm calls all you need? towards scaling laws of compound inference systems. *arXiv preprint arXiv:2403.02419*, 2024.

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

[6] 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.

[7] Ben Shneiderman and Catherine Plaisant. *Designing the user interface: strategies for effective human-computer interaction*. 2010.

[8] 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*, 2021.

[9] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[10] Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*, 2024.

[11] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

[12] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

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

[14] Shalev Lifshitz, Sheila A McIlraith, and Yilun Du. Multi-agent verification: Scaling test-time compute with multiple verifiers. *arXiv preprint arXiv:2502.20379*, 2025.

[15] Hangfan Zhang, Zhiyao Cui, Xinrun Wang, Qiaosheng Zhang, Zhen Wang, Dinghao Wu, and Shuyue Hu. If multi-agent debate is the answer, what is the question? *arXiv preprint arXiv:2502.08788*, 2025.

[16] Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. *arXiv preprint arXiv:2406.04692*, 2024.

[17] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR*, 2023.

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

[19] Hongwei Liu, Zilong Zheng, Yuxuan Qiao, Haodong Duan, Zhiwei Fei, Fengzhe Zhou, Wenwei Zhang, Songyang Zhang, Dahua Lin, and Kai Chen. MathBench: Evaluating the theory and application proficiency of LLMs with a hierarchical mathematics benchmark. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6884–6915, Bangkok, Thailand, 2024. Association for Computational Linguistics.

[20] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*, 2024.

[21] Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning, ICML*, 2024.

[22] Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *The Twelfth International Conference on Learning Representations, ICLR*, 2024.

[23] Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. Chateval: Towards better llm-based evaluators through multi-agent debate. In *The Twelfth International Conference on Learning Representations, ICLR*, 2024.

[24] Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. Language models are multilingual chain-of-thought reasoners. In *The Eleventh International Conference on Learning Representations, ICLR*, 2023.

[25] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

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

[27] Team Gemini, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[28] Anthropic. Introducing the next generation of claude. *www.anthropic.com/news/claude-3-family*, 2024.

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

[30] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

[31] Qineng Wang, Zihao Wang, Ying Su, Hanghang Tong, and Yangqiu Song. Rethinking the bounds of llm reasoning: Are multi-agent discussions the key? In *62nd Annual Meeting of the Association for Computational Linguistics, ACL 2024*, pages 6106–6131. Association for Computational Linguistics (ACL), 2024.

[32] Yasin Abbasi Yadkori, Ilja Kuzborskij, András György, and Csaba Szepesvári. To believe or not to believe your llm: Iterative prompting for estimating epistemic uncertainty. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 58077–58117. Curran Associates, Inc., 2024.

[33] Yashar Talebirad and Amirhossein Nadiri. Multi-agent collaboration: Harnessing the power of intelligent llm agents. *arXiv preprint arXiv:2306.03314*, 2023.

[34] Jintian Zhang, Xin Xu, Ningyu Zhang, Ruibo Liu, Bryan Hooi, and Shumin Deng. Exploring collaboration mechanisms for LLM agents: A social psychology view. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14544–14607, August 2024.

[35] Chen Qian, Zihao Xie, Yifei Wang, Wei Liu, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Scaling large-language-model-based multi-agent collaboration. *arXiv preprint arXiv:2406.07155*, 2024.

[36] Ziyu Wan, Yunxiang Li, Yan Song, Hanjing Wang, Linyi Yang, Mark Schmidt, Jun Wang, Weinan Zhang, Shuyue Hu, and Ying Wen. Rema: Learning to meta-think for llms with multi-agent reinforcement learning. *arXiv preprint arXiv:2503.09501*, 2025.

[37] Yiqun Zhang, Peng Ye, Xiaocui Yang, Shi Feng, Shufei Zhang, Lei Bai, Wanli Ouyang, and Shuyue Hu. Nature-inspired population-based evolution of large language models. *arXiv preprint arXiv:2503.01155*, 2025.

[38] OpenAI. Learning to reason with llms. *https://openai.com/index/learning-to-reason-with-llms*, 2024.

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

[40] Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2311.17311*, 2023.

[41] Swarnadeep Saha, Omer Levy, Asli Celikyilmaz, Mohit Bansal, Jason Weston, and Xian Li. Branch-solve-merge improves large language model evaluation and generation. *arXiv preprint arXiv:2310.15123*, 2023.

[42] Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. More agents is all you need. *arXiv preprint arXiv:2402.05120*, 2024.

[43] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in Neural Information Processing Systems*, 30, 2017.

[44] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

[45] Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, et al. Prometheus: Inducing fine-grained evaluation capability in language models. In *The Twelfth International Conference on Learning Representations*, 2023.

[46] Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, et al. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2, 2024.

[47] Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

[48] Sean Welleck, Jiacheng Liu, Ximing Lu, Hannaneh Hajishirzi, and Yejin Choi. Naturalprover: Grounded mathematical proof generation with language models. *Advances in Neural Information Processing Systems*, 35:4913–4927, 2022.

[49] Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, et al. From generation to judgment: Opportunities and challenges of llm-as-a-judge. *arXiv preprint arXiv:2411.16594*, 2024.

[50] Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. *arXiv preprint arXiv:2305.19118*, 2023.

[51] Andries Petrus Smit, Nathan Grinsztajn, Paul Duckworth, Thomas D Barrett, and Arnu Pretorius. Should we be going MAD? A look at multi-agent debate strategies for LLMs. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 45883–45905, 21–27 Jul 2024.

[52] Yunxuan Li, Yibing Du, Jiageng Zhang, Le Hou, Peter Grabowski, Yeqing Li, and Eugene Ie. Improving multi-agent debate with sparse communication topology. *arXiv preprint arXiv:2406.11776*, 2024.

[53] Kyungha Kim, Sangyun Lee, Kung-Hsiang Huang, Hou Pong Chan, Manling Li, and Heng Ji. Can llms produce faithful explanations for fact-checking? towards faithful explainable fact-checking via multi-agent debate. *arXiv preprint arXiv:2402.07401*, 2024.

[54] Dawei Li, Zhen Tan, Peijia Qian, Yifan Li, Kumar Satvik Chaudhary, Lijie Hu, and Jiayi Shen. Smoa: Improving multi-agent large language models with sparse mixture-of-agents. *arXiv preprint arXiv:2411.03284*, 2024.

[55] Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. Large language model routing with benchmark datasets. *arXiv preprint arXiv:2309.15789*, 2023.

[56] Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. Routing to the expert: Efficient reward-guided ensemble of large language models. *arXiv preprint arXiv:2311.08692*, 2023.

[57] Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. Routerbench: A benchmark for multi-llm routing system. *arXiv preprint arXiv:2403.12031*, 2024.

[58] Mohammed Muqeeth, Haokun Liu, Yufan Liu, and Colin Raffel. Learning to route among specialized experts for zero-shot generalization. *arXiv preprint arXiv:2402.05859*, 2024.

[59] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms from preference data. In *The Thirteenth International Conference on Learning Representations*, 2024.

[60] Tatsuki Kuribayashi, Yohei Oseki, Takumi Ito, Ryo Yoshida, Masayuki Asahara, and Kentaro Inui. Lower perplexity is not always human-like. *arXiv preprint arXiv:2106.01229*, 2021.

[61] Costas Mavromatis, Petros Karypis, and George Karypis. Pack of llms: Model fusion at test-time via perplexity optimization. *arXiv preprint arXiv:2404.11531*, 2024.

[62] Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press, 2017.

[63] Sina Aeeneh, Nikola Zlatanov, and Jiangshan Yu. New bounds on the accuracy of majority voting for multiclass classification. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2024.

[64] I. Csiszar. The method of types [information theory]. *IEEE Transactions on Information Theory*, 44(6):2505–2523, 1998.

A Detailed Experimental Setup

A.1 Benchmarks

The details of the six datasets we used are as follows:

- GSM8K [8]: A benchmark for evaluating grade school mathematical reasoning and problem-solving abilities, containing 1,319 questions.
- MATH [18]: A challenging dataset consisting of 500 high school and college-level math problems.
- MathBench [19]: A comprehensive dataset spanning a wide range of mathematical disciplines, designed to evaluate both theoretical understanding and practical problem-solving skills. We use the Arith subset, which contains 300 questions.
- MGSM [24]: A multi-language version of GSM8K, where we use 10 non-English languages and sample a total of 1,000 test samples.
- DATE [25]: A dataset designed to test symbolic reasoning capabilities through simple string manipulation tasks, containing 396 questions.
- MMLU-Pro [20]: A robust and challenging dataset for massive multitask understanding. We randomly select a subset of 500 questions for evaluation.

A.2 Hyperparameter Settings

We set the temperature and top_p of GPT-4o mini to 1 in all experiments, while all other LLMs were kept with their default hyperparameters.

The settings for external weights W_β in Sections 4.3 and 4.4 are shown in Tables 1 and 2, respectively.

Model	GSM8K	MATH	MathBench	MGSM	DATE	MMLU-Pro
GPT-4o mini	1	2	1	1	1.5	1
Gemini 1.5 Flash	1	2	1	1	1.5	1.5
Claude 3 Haiku	1	1	1	1	1	1

Table 1: Settings for external weights W_β in Section 4.3.

Model	MathBench	DATE
GPT-4o mini	2	1.5
Gemini 1.5 Flash	2	1.5
Claude 3 Haiku	2	1
Llama-3.1-8B-Instruct	1	1
Gemma-2-9B-It	1	1
Qwen2.5-7B-Instruct	1	1

Table 2: Settings for external weights W_β in Section 4.4.

A.3 Multi-Agent Debate Systems Setup

For both MAD and MAD (MLD), we allocate 3 rounds of debate, with a sampling budget of 5 per round. For ChatEval, we set the collaboration round to 5, with each round involving a general public, a critic, and a scientist. For AgentVerse, we set sampling budget to 17, consisting of 3 role assigners, 5 solvers, 3 critic 0 , 3 critic 1 and 3 evaluators. We set sampling budget of MOA to 16, consisting of 5 proposer layers with 3 proposers each and 1 aggregator layer, totaling 16 samples. For ModelSwitch, we allow three LLMs sample up to 5 times respectively.

B Difference Between Consistency and Perplexity

The term "consistency" in this paper differs from "perplexity" [60, 61] (where perplexity is a measurement of how well a language model predicts a sample, defined as the exponential of the average negative log-likelihood of the predicted token probabilities). Instead, "consistency" refers to the agreement of generated answers produced by the model across multiple samples for queries with definitive answers. For example, the model may generate two samples with completely different reasoning processes but arrive at the same answer. In this case, two samples may have significantly different perplexities, but we consider their answers to be consistent.

C Theoretical Analysis for the Whole Dataset

C.1 Set Up and Notations

To theoretically understand the promotion of the multi-LLM repeated sampling method, we first introduce a simple formulation of our problem. We start with one single query q and a given model M. Let $M(q)$ denote the set of all possible answers of model M on query q and $\Delta(M(q))$ as the distribution of M's answer given query q. In the following context, we denote $\Delta(M(q))$ as $\Delta(M)$ for simplicity. Then we denote Φ as the algorithm for determining the final answer, such as majority voting. Let $\Phi(M)$ denote the distribution of model's final answers after applying Φ (we omit the query q). We denote $f : \Delta(M) \to \mathbb{R}$ as the loss function. By selecting f as the 0-1 loss, we can evaluate the model's performance by the probability of it generating the correct answer (we always denote A as the correct answer in the following context). Then the performance of model M could be evaluated by $\mathbb{E}\big[f(\Phi(M)\big]$, where the expectation is over the query distribution. To verify that using expectation to measure performance is reasonable, we have compared our theoretical results with the experimental results on the dataset MathBench with $K = 10, 16$. The results exhibit **complete consistency** in terms of ordinal relationships and demonstrate substantial agreement at the numerical level.

Without loss of generality, we consider the case of two models. Let MV denote the majority voting algorithm with $2K$ samples for single model, and MS denote our ModelSwitch Algorithm 1 samples K times for model M_1 and model M_2, respectively. The performance of the these algorithms can be denoted as $\mathbb{E}\big[f(\text{MS}(M_1, M_2))\big]$, $\mathbb{E}\big[f(\text{MV}(M_1))\big]$ and $\mathbb{E}\big[f(\text{MV}(M_2))\big]$.

In the following analysis, we show that under certain conditions, the performance of ModelSwitch can outperform single model repeated sampling then majority voting, i.e.,

$$\mathbb{E}\big[f(\text{MS}(M_1, M_2)\big] < \min\Big\{\mathbb{E}\big[f(\text{MV}(M_1))\big], \mathbb{E}\big[f(\text{MV}(M_2))\big]\Big\}. \tag{4}$$

C.2 Two Intuitive Examples

We provide two examples to aid understanding. These examples demonstrate two potential reasons why the MS method can enhance performance. We always let f be the $0-1$ loss in this section.

Example 1. *Consider a problem set that consists of one single problem q with the correct answer A, and two Model M_1, M_2; And M_1, M_2's (one sample) answer's distribution are respectively* $(0.4, 0.6, 0)$ *and* $(0.4, 0, 0.6)$ *(they share the same answer space $\{A, B, C\}$). If we let the number of samplings K be sufficiently large, by the Law of Large Numbers, we have*

$$f(\mathit{MV}(M_i)) \to 1 \quad (i = 1, 2); \quad f(\mathit{MS}(M_1, M_2)) \to 0, \quad \text{as } K \to +\infty$$

Clearly, Equation (4) *holds.*

We remark that, on single problem, this phenomenon does not exist in the case of binary classification.

Example 2. *Consider a problem set that consists of two multi-class classification problems q_1, q_2 with the correct answer A, and two Model M_1, M_2; And M_1, M_2's (one sample) answer's distribution of the two problems are respectively: M_1 : $(0.90, 0.05, 0.05)_{q_1}, (0.10, 0.50, 0.40)_{q_2}$ and M_2 : $(0.10, 0.50, 0.40)_{q_1}, (0.90, 0.05, 0.05)_{q_2}$, (they share the same answer space $\{A, B, C\}$).*
One could verify that for any M, ModelSwitch would outperform single model in this case, although M_1, M_2 has same loss on this problem set.

This example illustrates that disparities in expertise and uncertainty on single error create opportunities for model ensemble methods to leverage their potential.

C.3 Understanding the Enhancement

We highlight that our analysis is not confined to binary classification problems, which inherently introduces significant hardness. To the best of our knowledge, theoretical analysis of majority voting in multi-class settings are exceedingly rare, existing approach is to apply Poisson approximation [62] to derive an upper bound [63].

Informally, we first define two events: (1) $\Omega_1 := \{f(\text{MV}(M_1)) < f(\text{MS}(M_1, M_2)\}$ denotes the event that ModelSwitch underperforms majority voting on this query; (2) $\Omega_2 := \{f(\text{MV}(M_1)) > f(\text{MS}(M_1, M_2)\}$ denotes the event that ModelSwitch outperforms Majority Voting on this query. Then let $\alpha = \mathbb{P}(\Omega_1), \beta = \mathbb{P}(\Omega_2)$ denote the probability of these two events respectively. The distribution of the count of each answer among the K answers follows a multinomial distribution across K trials, with a probability of $\Delta(M)$. Let x_A denote the number of times the right answer appears in K samples.

Accordingly, we define $C^\epsilon(M_i)$ as the probability that M_i generates the correct answer with a frequency of at least $(1-\epsilon)$: $C^\epsilon(M_i) = \mathbb{P}(x_A \geq (1-\epsilon)K))$. Finally, we denote $W^\epsilon(M_i)$ as an upper bound of the probability of M_i presenting at least $(1-2\epsilon)$ frequency on a wrong answer. Using the methods of types Lemma C.3, we can calculate that $W^\epsilon(M_i) = (2K\epsilon)^{n+1} \cdot \{2^{K((1-2\epsilon)\log(\max_{j\neq A}\Delta(M_i)_j + \epsilon_0)}\}$.

The following theorem captures this intuitive insight: on the subset of queries where the performance discrepancy between two models is non-negligible, if the superior model is sufficiently confident in the correct answer on average – which is measured by $\mathbb{E}[C^\epsilon(M_i)|\Omega_i]$, while the inferior model exhibits adequate uncertainty among different incorrect answers – which is measured by $(\mathbb{E}[1 - W^\epsilon(M_{3-i})|\Omega_i])$, then the ModelSwitch algorithm can achieve performance surpassing single model.

More specifically, we divide the queries into two parts:(1) part I is queries that Model 2 could 'enhance' Model 1's performance; (2) part II is queries that Model 2 'mislead' Model 1. Then, if the total 'enhance' effect is stronger than the total 'mislead' effect, ModelSwitch will outperform single model. The condition that total 'enhance' effect is stronger than the total 'mislead' effect can be denoted as:

$$\begin{aligned}&\beta \cdot \Big(\mathbb{E}[C^\epsilon(M_2)|\Omega_2] \cdot \mathbb{E}[1 - W^\epsilon(M_1)|\Omega_2] - \mathbb{E}[1 - f(\text{MV}(M_1))|\Omega_2]\Big) \\ &\qquad > \alpha \cdot \Big(\mathbb{E}[C^\epsilon(M_1)|\Omega_1] \cdot \mathbb{E}[W^\epsilon(M_2)|\Omega_1] + \mathbb{E}[1 - C^\epsilon(M_1)|\Omega_1]\Big),\end{aligned} \tag{5}$$

where β, α respectively reflects the proportion of part I and part II, while the term after β measures the average 'enhance' effect and the term after α measures the average 'mislead' effect. The detailed derivation of this condition can be found in Appendix C.5. Then we have the following theorem:

Theorem C.1. *If there exists some $\epsilon \in [0, 0.25)$ such that the condition Equation (5) holds, we have*

$$\mathbb{E}\big[f(\mathit{MS}(M_1, M_2)\big] < \min\Big\{\mathbb{E}\big[f(\mathit{MV}(M_1))\big], \mathbb{E}\big[f(\mathit{MV}(M_2))\big]\Big\} \tag{6}$$

C.4 Validation of Theoretical Formulation

In this subsection, we verify whether our formulation is reasonable. We set f as $0-1$ loss and estimate the query distribution by empirical mean. That is, for each query q in the problem set, we query M_i for 50 times independently. We use $p_q^{M_i}$ denote the answer distribution of model M_i. Then use the data collected to estimate $p_q^{M_i}$ by the empirical distribution $\hat{p}_q^{M_i}$. And we view the query distribution as the uniform distribution over the whole MathBench [19] problem set. In practice, we find that for each problem q, the total types of different answers that appear more than 1 time is no more than 6. So, we set $M(q) = \{A, B, C, D, E, F\} := [A, F]$. Then, we calculate

$$\frac{1}{N}\sum_{n=1}^{N}\Big(\sum_{\substack{x_A = \max x_j \\ x_A + \ldots + x_F = M}} \frac{K!}{x_A! \ldots x_F!} \prod_{j\in[A,F]} \hat{p}_{q_n}^{M_i}(j)^{x_j}\Big), \qquad i = 1, 2;$$

and

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{(\boldsymbol{x},\boldsymbol{y})\in R}\Big(\frac{K!}{x_A! \ldots x_F!}\Big) \cdot \Big(\frac{K!}{y_A! \ldots y_F!}\Big) \prod_{j\in[A,F]} \big(\hat{p}_{q_n}^{M_1}(j)^{x_j} \hat{p}_{q_n}^{M_2}(j)^{y_j}\big),$$

where $R = \{(\boldsymbol{x},\boldsymbol{y}) | x_A + y_A = \max_j(x_j + y_j), \sum_j x_j = \sum_j y_j = K\}$. Those two expressions respectively estimate the performance of the two single model and 2-LLM repeated sampling. The results are reported in Figure 8.

C.5 Proof of Theorem 5.1

Definition C.2 (Empirical distribution, type class)**.** Let $\boldsymbol{x} = \{x_1, \ldots, x_K\}$ with $x_i \in \mathcal{X} = \{1, \ldots, r\}$ and

$$N(a) = \sum_{i=1}^{K} \mathbb{1}\{x_i = a\}, \text{P}(a) = \frac{N(a)}{K}.$$

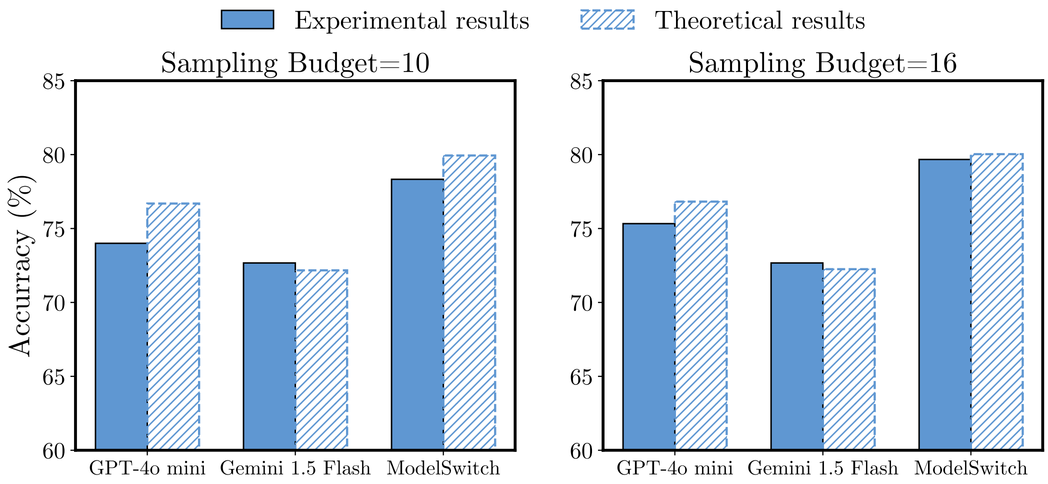

Figure 8: Comparison of the theoretical results and the experimental results on MathBench, with sampling budgets $K = 10, 16$. The theoretical results are in substantial agreement with the actual experimental results.

The empirical distribution of $\boldsymbol{x}$ is denoted by $\mathrm{P}_{\boldsymbol{x}} = (\mathrm{P}(1), \ldots, \mathrm{P}(r))$. Let $\mathbb{P}_K$ denote the collection of all empirical distributions of sequences of length K, i.e., $\mathbb{P}_K = \{\mathrm{P}_{\boldsymbol{x}} : \boldsymbol{x} \in \mathcal{X}^K\}$. For any $\mathrm{P} \in \mathbb{P}_K$, the **type class** or **type** of P is denoted by $T(\mathrm{P}) = \{\boldsymbol{x} : \mathrm{P}_{\boldsymbol{x}} = \mathrm{P}\}$. The type class of $\boldsymbol{x}$ is $T_{\boldsymbol{x}} = T(\mathrm{P}_{\boldsymbol{x}}) = \{\tilde{\boldsymbol{x}} : \mathrm{P}_{\tilde{\boldsymbol{x}}} = \mathrm{P}_{\boldsymbol{x}}\}$. Define $\mathbb{Q}$ is a probability density function, $\mathbb{Q} = \{Q(x)\}_{x \in \mathcal{X}}$. Let $Q(\boldsymbol{x}) = \prod_i Q(x_i)$ and for any $S \subset \mathcal{X}^K$,

$$\mathbb{Q}(S) := \sum_{\boldsymbol{x} \in S} Q(\boldsymbol{x}).$$

Lemma C.3 (The Method of Types). *For $\forall$ p.d.f $\mathbb{Q}$ and $\forall P \in \mathbb{P}_K$,*

$$\frac{1}{(K+1)^{r-1}} 2^{-K \cdot D_{KL}(P||\mathbb{Q})} \leq \mathbb{Q}(T(P)) \leq 2^{-K \cdot D_{KL}(P||\mathbb{Q})},$$

where $D_{KL}(\cdot||\cdot)$ is the KL-divergence.

Proof. Please refer to [64] for detailed proofs. □

Proof of Theorem 5.1

Proof: It is equivalent to demonstrating that the probability of ModelSwitch choosing the correct answer is higher than that of a single model. When f is selected as $0-1$ loss, it is equivalent to show that the probability of ModelSwitch selects the correct answer (we denote it as A) is higher than Majority Voting on single model. Specifically, aligned with our experiment, we always set the model with lower $\mathbb{E}[f(\Phi(M))]$ to be the first model in ModelSwitch Algorithm. Thus, Equation (4) reduces to

$$\mathbb{E}[f(\text{MS}(M_1, M_2))] < \mathbb{E}[f(\text{MV}(M_1)].$$

By the linearity of expectation, we have

$$\begin{aligned} &\mathbb{E}[f(\text{MS}(M_1, M_2)] - \mathbb{E}[f(\text{MV}(M_1)] \\ &= P(\Omega_1) \cdot \mathbb{E}[f(\text{MS}(M_1, M_2)) - f(\text{MV}(M_1)] + P(\Omega_2) \cdot \mathbb{E}[f(\text{MS}(M_1, M_2)) - f(\text{MV}(M_1)]. \end{aligned} \tag{7}$$

Recall that $\Omega_1 := \{f(\text{MV}(M_1)) < f(\text{MS}(M_1, M_2)\}$, $\Omega_2 := \{f(\text{MV}(M_1)) > f(\text{MS}(M_1, M_2)\}$, we know that the first term in Equation (7) is positive and the second term is negative. Next we bound them respectively.

Denote $G_K^\epsilon(M_i)$ as the event that among more than $(1-\epsilon)K$ samples, model M_i has chosen the correct answer. Recall that $C^\epsilon(M_i) = \mathbb{P}(x_A \geq (1-\epsilon)K) = P(G_K^\epsilon(M_i))$. Under the event $G_K^\epsilon(M_i)$, Model Switch ($2K$ samples) will always give the correct answer if the other model $M_{i'}(i' \neq i)$ generates the incorrect answers less than $(1-2\epsilon)K$ times. Moreover, by methods of types, we can derive that

$$\begin{aligned}\forall j \neq A,\;\; P(x_j^{M_{i'}} \geq (1-2\epsilon)K) &= \sum_{\{T(P): x_j \in [1-2\epsilon K, K]\}} p^{M_j}(T(P)) \\ &\leq \sum_{\{T(P): x_j \in [1-2\epsilon K, K]\}} 2^{-K \cdot D_{KL}(P || p^{M_{i'}})} \\ &\leq \sum_{\{T(P): x_j \in [1-2\epsilon K, K]\}} 2^{-K \cdot [(1-2\epsilon)\log(\frac{1}{p_j}) - \epsilon_o]} \\ &\leq (2\epsilon K)^{n+1} \cdot 2^{-K \cdot [(1-2\epsilon)\log(\frac{1}{p_j}) - \epsilon_o]},\end{aligned}$$

where $n = \max_{i \in \{1,2\}} \max_q |\mathcal{A}_q^{M_i}|$, $\epsilon_0 = 2\epsilon(\log(2\epsilon) - \log(n-1))$. The second inequation follows from Lemma C.3, and the third inequation is by the definition of KL divergence and the last inequation is follows from the fact that eligible types are no more than $(2\epsilon K)^n \cdot (2\epsilon K)$. Hence, we have

$$P(\{\max_{j \neq A}\{x_j\} \geq (1-2\epsilon)K\}) \leq (2\epsilon K)^{n+1} \cdot 2^{K \cdot [(1-2\epsilon)\log(\max_{j \neq A} p_j) + \epsilon_o]}.$$

Therefore, we have

$$\begin{aligned}f(\text{MS}(M_1, M_2)) - f(\text{MV}(M_1)) &\leq (1 - P(G_K^\epsilon(M_1))) + P(G_K^\epsilon(M_1)) \cdot P(\{\max_{j \neq A}\{x_j^2\} \geq (1-2\epsilon)K\}) \\ &\leq (1 - C_K^\epsilon(p^{M_1})) + C_K^\epsilon \cdot (2\epsilon K)^{n+1} \cdot 2^{-K \cdot [(1-2\epsilon)\log(\frac{1}{p_j}) - \epsilon_o]} \\ &= (1 - C_K^\epsilon(p^{M_1})) + C_K^\epsilon \cdot W_K^\epsilon(p^{M_2}); \qquad (8) \\ f(\text{MS}(M_1, M_2)) - f(\text{MV}(M_1)) &\leq 1 - P(G_K(M_2)) \cdot [1 - P(\{\max_{j \neq A}\{x_j^1\} \geq (1-2\epsilon)K\})] - f(\Phi_{MV}^{2K}(p^{M_1})) \\ &\leq -C_K^\epsilon(p^{M_2}) \cdot [1 - W_K^\epsilon(p^{M_1})] + (1 - f(\Phi_{MV}^{2K}(p^{M_1})). \qquad (9)\end{aligned}$$

Note that we explicitly leverage the pattern $E[XY] = E[X \cdot E[Y|X]] \leq E[X] \cdot E[Y]$ in the last step, this inequality holds when X, Y are negatively correlated or $E[Y|X]$ is decreasing with respect to X. This expectation decomposition inequality is natural, since the more confidence M_1 puts on one single incorrect answer, the more likely M_2 could promote it's performance on this question.

By choosing an ϵ that satisfies Equation (5), and combining Equations (7) and (9), we have completed the proof of this theorem.