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

Jianhao Chen[1 2*], **Zishuo Xun**[2 3*], **Bocheng Zhou**[2*], **Han Qi**[2*], **Hangfan Zhang**[4*], **Qiaosheng Zhang**[2], **Yang Chen**[2], **Wei Hu**[1], **Yuzhong Qu**[1†], **Shuyue Hu**[2†]

[1]State Key Laboratory for Novel Software Technology, Nanjing University
[2]Shanghai Artificial Intelligence Laboratory
[3]The University of Auckland
[4]The Pennsylvania State University
jhchen.nju@gmail.com, zxun233@aucklanduni.ac.nz, hbz5148@psu.edu, {whu,yzqu}@nju.edu.cn,
{zhoubocheng,qihan,zhangqiaosheng,chenyang,hushuyue}@pjlab.org.cn

Abstract

This paper presents a simple, effective, and cost-efficient strategy, named ModelSwitch, to improve LLM performance by scaling test-time compute. ModelSwitch 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 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, our strategy 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.

Code — https://github.com/JianhaoChen-nju/ModelSwitch

Introduction

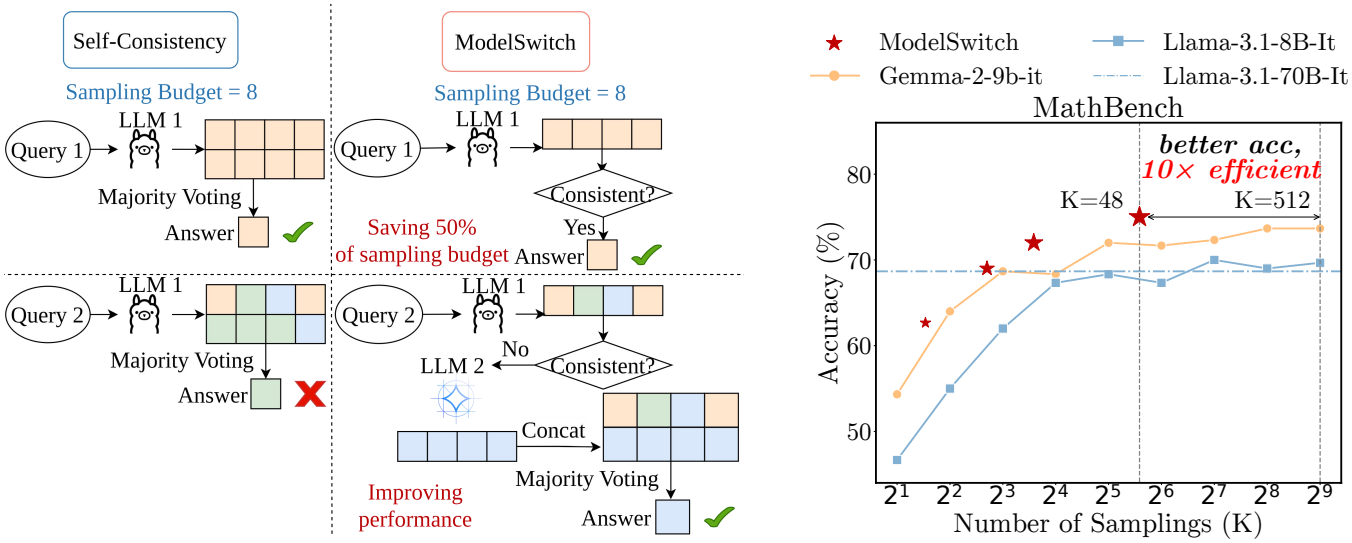
Scaling has been a major driving force of recent rapid advancements in large language models (LLMs). While training-time compute (Rae et al. 2021) scaling is reaching limits, inference-time compute scaling offers a promising alternative (Snell et al. 2024). An emerging direction is to scale inference-time compute based on the *generation-verification* paradigm. Here, an LLM is queried multiple times to produce various candidate answers, which are then verified to yield a final response. Studies across various LLMs and benchmarks consistently demonstrate that simply scaling the number of generated samples significantly improves the *coverage* of correct answers (Brown et al. 2024). This has led to recent efforts pushing sample counts into the hundreds or thousands (Brown et al. 2024; Chen et al. 2024a; Liu et al. 2025) to improve answer correctness.

However, *do we truly need so many samples*? Scaling repeated sampling is computationally expensive, with the consumption of floating point operations increasing linearly with the number of samplings (Kaplan et al. 2020). Additionally, it causes significant delays, which impacts user experience (Shneiderman and Plaisant 2010). 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 (Cobbe et al. 2021; Yang et al. 2024; Zhang et al. 2024b) and automatic verifiers (Li et al. 2022; Schick et al. 2023) have been proposed, whereas LLM-as-a-judge (Zheng et al. 2023; Lifshitz, McIlraith, and Du 2025) has also been extensively explored.

Orthogonal to recent efforts, we focus on the generation side. We believe different LLMs, trained on varied data and with distinct approaches, possess complementary strengths. Even on the same tasks, two general-purpose LLMs might excel at different types of questions (Zhang et al. 2025a; Wang et al. 2024a). 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.

We evaluate ModelSwitch on diverse datasets covering knowledge, reasoning, and domain-specific challenges. Using Gemini 1.5 Flash and GPT-4o mini, ModelSwitch sig-

*This work was done during the author's intership at Shanghai Arificial Intelligence Laboratory.
†Corresponding author.



(a) (b)

Figure 1: (a) An overview of how ModelSwitch works between two LLMs. (b) Performance comparison of ModelSwitch (MS) and self-consistency (SC) (Wang et al. 2023) on MathBench (Liu et al. 2024). MS switches between Gemma-2-9B-It and Llama-3.1-8B-Instruct. The curves show SC performance for individual LLMs, while horizontal lines represent single-sample baselines of larger LLM Llama-3.1-70B-Instruct. MS achieves 75% accuracy (with 48 samples), outperforming SC using Gemma-2-9B-It (73.7%, 512 samples) at 10× efficiency. Additionally, combining 9B and 8B models achieves (69%) accuracy comparable to a 70B model (68.7%) with only 7 samples.

nificantly outperforms the self-consistency (Wang et al. 2023) performance of each respective LLM and even surpasses advanced LLMs like GPT-4o and Gemini 1.5 Pro—achievements unattainable by self-consistency. Additionally, ModelSwitch demonstrates superior sample efficiency, reducing average sampling per query by 34% across all datasets, saving computational costs while maintaining high performance. Furthermore, ModelSwitch achieves state-of-the-art results on four datasets, outperforms five multi-agent or multi-LLM methods, and ranks second on the other two datasets. Notably, ModelSwitch achieves 63.2% accuracy on MMLU-Pro (Wang et al. 2024c), surpassing the best single LLM by 10.2 points and outperforming other methods, including MAD (Du et al. 2024) (47.6%), AgentVerse (Chen et al. 2024b) (44.8%), ChatEval (Chan et al. 2024) (44%), ReConcile (Chen, Saha, and Bansal 2024) (49.6%), MAD (multi-LLM version) (44%), and MOA (Wang et al. 2024a) (52.6%). Additional experiments show ModelSwitch can integrate with verification methods like reward models for further performance gains.

We formally analyze ModelSwitch's improvements over self-consistency. Our findings provide sufficient and necessary conditions for ModelSwitch (using two models) to outperform single-model self-consistency. We also establish a theoretical bound on its efficiency gains, showing that if the probability of each model generating consistent answers exceeds a constant $c > 0$ and models are sampled equally, the expected sampling count decreases by a factor of $\frac{1}{n \times c}$, 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 providing a deeper understanding of why multiple-LLM generation surpasses single-LLM generation.

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 (Wang et al. 2023) 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

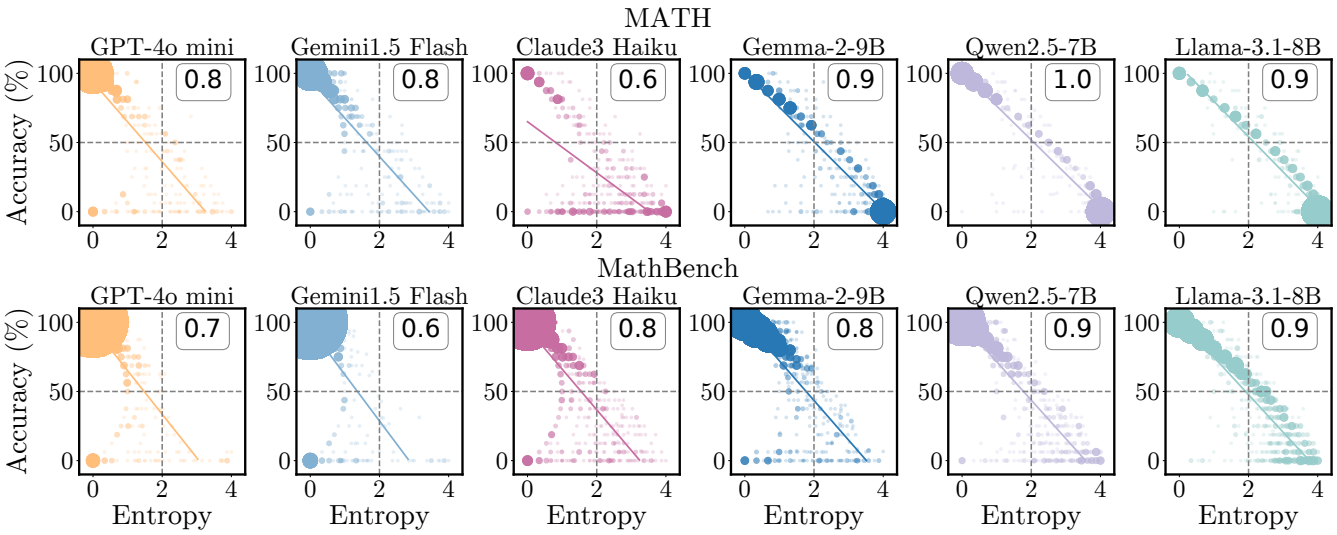


Figure 2: Correlation between consistency (entropy) and accuracy of answers from six LLMs on MATH and MathBench. We use the transparency and size of the scatter points to indicate the number of queries corresponding to each point (the larger and more opaque the point, the greater the quantity). We report 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 ($p < 0.001$).

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.

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 *consistent implies correct, while chaotic implies wrong*.

ModelSwitch: Harnessing Consistency in Multi-LLM Generation

ModelSwitch is a simple yet effective framework for multi-LLM repeated sampling that simultaneously achieves high performance and efficiency. Our core idea leverages the strong correlation between consistency and accuracy during repeated sampling: When a model's answers are consistent, we can confidently reduce sampling, relying on the current answer. When a model generates inconsistent answers, it signals the model "does not know." Instead of continuing with the same model, ModelSwitch switches to another, embracing the possibility that "the next model you encounter might know what the previous one did not." This approach harnesses complementary strengths across models, optimizing both efficiency and accuracy. Our ModelSwitch design is detailed in Algorithm 1.

ModelSwitch replaces a single LLM with multiple LLMs

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($all\_results$)

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. Specifically, given a 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 To account for performance differences among models, we use a weighted voting algorithm with internal (W_α) and external (W_β) weights to aggregate answers A from all models M.

$$MS(q) = \underset{a \in A}{\operatorname{argmax}} \sum_{i=1}^{M} \mathrm{W}_\alpha \times \mathrm{W}_\beta \times \mathrm{count}(a, A_i) \quad (1)$$

Internal Weights: Measure a model's confidence based on answer consistency. Calculated via entropy and normalized to $[\text{bias}, 1]$, where $\text{bias} = \frac{1}{\text{len(samples)}}$. Higher consistency yields a higher weight. Example: For answer sets $\mathrm{M}_1 = \{\mathrm{A} \times 3, \mathrm{B} \times 2\}$, $\mathrm{M}_2 = \{\mathrm{B} \times 3, \mathrm{A} \times 2\}$, $\mathrm{M}_3 = \{\mathrm{C} \times 5\}$, M_3's uniformity gives C a higher score.

$$\mathrm{W}_\alpha = \text{bias} + (1 - \text{bias}) \left(1 - \frac{\text{Entropy}}{\text{norm}}\right), \text{if norm} > 0 \text{ else } 1 \quad (2)$$

External Weights: Assess prior performance, ensuring models with significantly different performances have corresponding importance.

Special Case Two-LLM switch is a very special case, as shown in Figure 1a, 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: (1) If $f(A_1) < \frac{K}{2}$, the samples of both models are considered. (2) 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.

Experiments: On the Efficacy, Efficiency and Scalability

Our experiments aim to answer the following four questions: (i) Does ModelSwitch outperform single-LLM repeated-sampling-then-voting in terms of efficacy and efficiency? (ii) Can ModelSwitch surpass other multi-agent (LLM) debate methods? (iii) Can ModelSwitch be extended with stronger verification methods?

Experimental Setup

Benchmarks We mainly evaluate our method on the following datasets that cover a wide range of knowledge, reasoning, and domain-specific challenges, providing a robust evaluation for our method: GSM8K (Cobbe et al. 2021), MATH (Lightman et al. 2023), MathBench (Liu et al. 2024), MGSM (Shi et al. 2023), DATE (Wei et al. 2022), MMLU-Pro (Wang et al. 2024c).

Models We primarily consider three lightweight, closed-source LLMs: GPT-4o mini, Gemini 1.5 Flash, and Claude 3 Haiku. Firstly, we compare ModelSwitch to self-consistency, using GPT-4o mini and Gemini 1.5 Flash, with GPT-4o and Gemini 1.5 Pro as additional baselines. Then, we use all three lightweight LLMs to evaluate ModelSwitch against other multi-agent debate methods. Finally, we explore enhancing ModelSwitch with stronger validation using Qwen2.5-MATH-RM-72B.

ModelSwitch Outperforms Self-Consistency 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 Flash) and ModelSwitch using both, under the same sampling budget. We also compared two self-consistency variants, Adaptive-Consistency (Aggarwal, Yang, and Mausam 2023) and Early-Stop-Consistency (Li et al. 2024d), which focus on reducing its overhead. Additionally, we considered CISC (Taubenfeld et al. 2025), but it's unsuitable for closed-source models due to its reliance on internal model logits. All our queries are asked in COT (Wei et al. 2022) format. The results are shown in Figure 3.

Efficacy Across all the benchmarks considered, ModelSwitch with two LLMs outperforms best-performing single LLM self-consistency. 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. Compared to Adaptive-Consistency and Early-Stop-Consistency, ModelSwitch also boasts a clear accuracy advantage overall.

Efficiency While ModelSwitch outperforms self-consistency, it also requires fewer samples. Taking the sam-

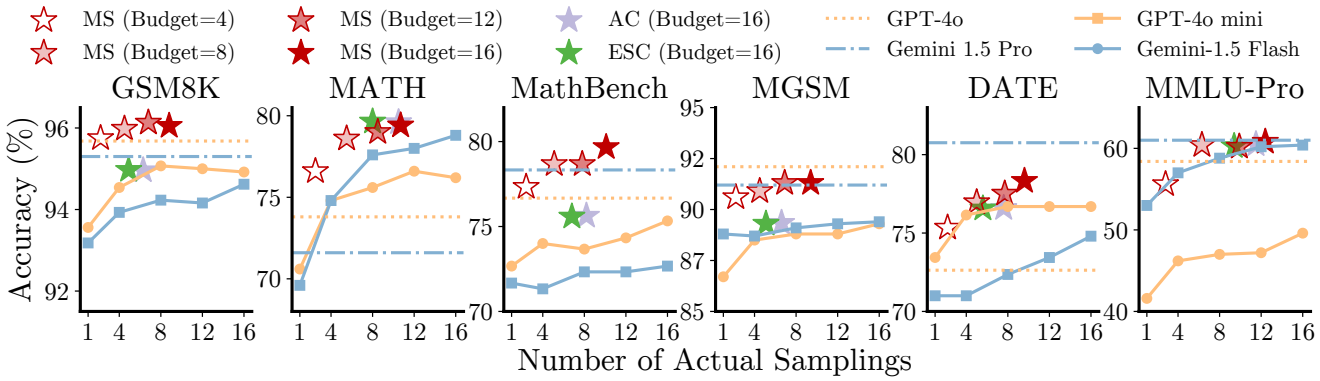


Figure 3: Performance comparison of self-consistency for each LLM (GPT-4o mini and Gemini 1.5 Flash) and ModelSwitch using both. We use horizontal lines to mark the single-sample results of more advanced LLMs GPT-4o and Gemini 1.5 Pro. The shade of red stars differentiates the sampling budgets of ModelSwitch. Purple and green stars show the performance of Adaptive-Consistency (AC) and Early-Stop-Consistency (ESC) of best-performing single LLM. Budget of 16 maximizes their accuracy without significantly increasing overhead.

pling 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.* Moreover, ModelSwitch is equally competitive with Adaptive-Consistency and Early-Stop-Consistency in terms of efficiency.

Scalability ModelSwitch, leveraging GPT-4o mini and Gemini 1.5 Flash, demonstrates superior scalability over self-consistency, Adaptive-Consistency and Early-Stop-Consistency for each LLM. It even outperforms more advanced models like GPT-4o and Gemini 1.5 Pro. *For instance, ModelSwitch surpasses both on GSM8K with an average of just 2.2 samples per query, and similarly on MathBench with 5.1 samples—achievements they can't match.*

ModelSwitch Better Leverages Multi-Agent Synergies than Debate Methods

In this section, we compare our approach to competitive multi-agent debate methods: MAD (Du et al. 2024), a strong baseline using inter-agent critique to enhance reasoning and factuality; ChatEval (Chan et al. 2024), which extends MAD to mimic human evaluation in complex tasks; AgentVerse (Chen et al. 2024b), emphasizing dynamic group composition and adaptive collaboration; ReConcile (Chen, Saha, and Bansal 2024) and MOA (Wang et al. 2024a), featuring hierarchical architectures for mutual evaluation among agents, seen as implicit debate. We additionally construct a new baseline, referred to as MAD (MLD), by extending MAD into a multi-LLM debate version. For MAD and AgentVerse, we use GPT-4o mini, while ChatEval and MAD (MLD) use GPT-4o mini and Gemini 1.5 Flash. ReConcile, MOA and ModelSwitch utilize GPT-4o mini, Gemini 1.5 Flash, and Claude 3 Haiku. To ensure fairness, all systems share a unified sampling budget of 15.

Efficacy As shown in Figure 4, our method achieves state-of-the-art performance on four datasets, with a significant lead on MMLU-Pro. For example, MAD (47.6%), AgentVerse (44.8%), ChatEval (44%), ReConcile (49.6%), MLD (44%) and MOA (52.6%) 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 (Zhang et al. 2025a; Wang et al. 2024b), 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.

ModelSwitch Can be Combined with Stronger Verification For Performance Boost

In this section, we integrate ModelSwitch with Qwen2.5-MATH-RM-72B, a powerful mathematical reward model, to score samples from GPT-4o mini and Gemini 1.5 Flash. Using the Best of N (BoN) strategy, we select the highest-scoring answer instead of relying on voting. Specifically, we replace the voting algorithm in Algorithm 1 with BoN. Performance curves comparing the best single LLM and ModelSwitch under the RM-BoN strategy are shown in Figure 5.

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).

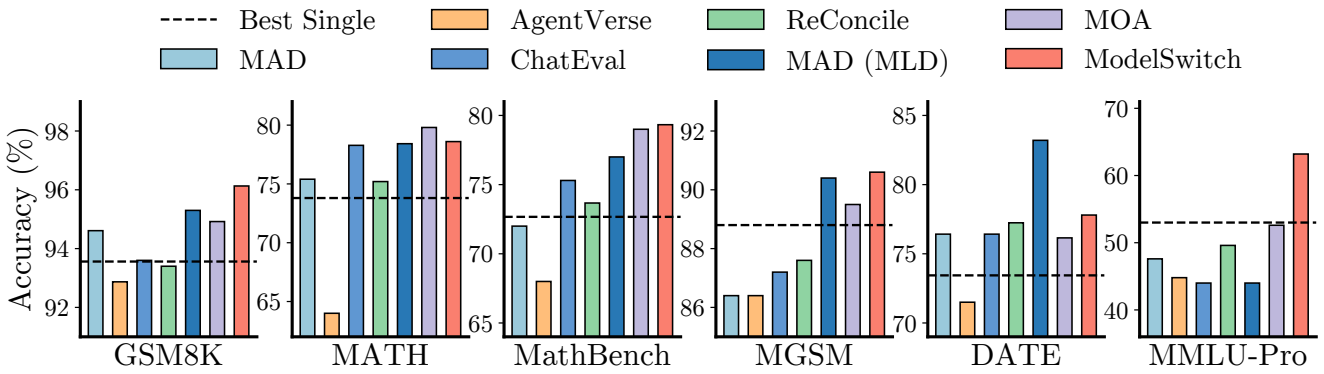


Figure 4: Performance comparison of different multi-agent debate systems under the same budget of 15 samples relative to the accuracy of the best single LLM with one sample. ModelSwitch achieves the best results on four datasets and the second-best results on the other two.

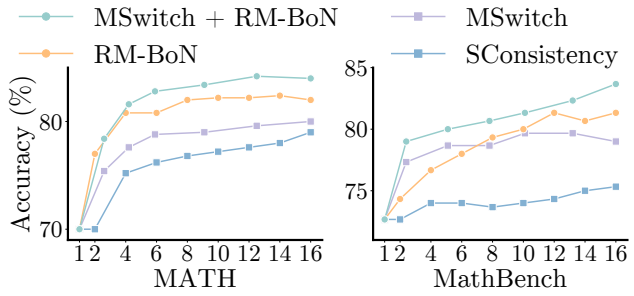


Figure 5: Performance comparison of ModelSwitch and single LLM combined with Qwen2.5-MATH-RM-72B as the reward model using the Best of N strategy.

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, ModelSwitch primarily emphasizes strategies in the generation (sampling) phase, thus it can be further enhanced by stronger selection (verification) methods to improve overall performance.

Understanding Why ModelSwitch Outperforms Self-Consistency

In this section, we analyze when ModelSwitch can improve performance for a given fixed query. 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.

Assuming equal sampling counts for two models, we focus on expected performance by comparing two probability vectors to assess the difference between mixed models and a single model. The probability of ModelSwitch providing a correct answer is $P = x_1 + x_2$. If P is the highest in the probability vector $p_1 + p_2$, ModelSwitch can provide the correct answer in expectation. We exclude cases where both M_1 and M_2 give the same final answer, whether correct or incorrect, as ModelSwitch doesn't alter the outcome in these scenarios. We present the following proposition.

Proposition 0.1. *Let $Q = y_1 - y_2$ and the number of options is denoted as m. The necessary condition for ModelSwitch to obtain the correct answer is $P > \frac{2}{m}$. The sufficient condition is*

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

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 0.1 can be explained intuitively: P should be as large as possible, as it directly determines whether ModelSwitch can provide the correct answer. Meanwhile, $|Q|$ should not be too large, as this would cause at least one inequality in the sufficient condition to fail. A large $|Q|$ indicates that the probability of an incorrect answer from M_1 or M_2 dominates, making it harder

to improve performance through mixing. This proposition doesn't require M_1 or M_2 to individually provide the correct final answer, meaning x_1 or x_2 don't need to be the largest probabilities. Even if both models fail to give the correct final answer, ModelSwitch can offset errors and still succeed, explaining why repeated sampling with multiple LLMs can outperform a single LLM.

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 $\frac{K}{n}$ times. Each model M_i has a probability P_i of producing a completely consistent answer list, where $P_n = 1$ indicates that if the first $n-1$ models do not produce a consistent answer, the n-th model must be sampled. In this scenario, the **expected actual number of samplings** ($\mathbb{E}[N_{\text{call}}]$) can be bounded as follows:

$$\mathbb{E}[N_{\text{call}}] = \frac{K}{n} \sum_{i=1}^{n} \left(\prod_{j=1}^{i-1} (1 - P_j) \right) P_i \tag{3}$$

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 $\frac{K}{n \times c}$, which reduces the number of samplings by a factor $\frac{1}{n \times c}$.

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 (Brown et al. 2024; Yadkori et al. 2024), then selecting the most suitable ones using verification mechanisms (Cobbe et al. 2021; Li et al. 2022). Multi-agent collaboration (Talebirad and Nadiri 2023; Zhang et al. 2024a; Qian et al. 2024; Wan et al. 2025; Zhang et al. 2025b) 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 (Wang et al. 2023; Li et al. 2024b; Brown et al. 2024) 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 and Deepseek-R1 have shown notable improvements when using self-consistency as a test-time compute strategy. Recent advances enhance this paradigm through structured decomposition: Universal Self-Consistency (Chen et al. 2023) directly generates final answers from N-best candidates via LM prompting, Adaptive-Consistency (Aggarwal, Yang, and Mausam 2023) and Early-Stop-Consistency (Li et al. 2024d) reduce overhead by designing early stopping mechanisms, while CISC (Taubenfeld et al. 2025) improves the accuracy of self-consistency by weighting answers based on internal model confidence. 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* (Christiano et al. 2017; Stiennon et al. 2020; Zhang et al. 2024b; Cui et al. 2025; Liu et al. 2025) 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) (Li et al. 2022; Schick et al. 2023) or mathematical proof-checking (Welleck et al. 2022) for rigorous validation; *LLM-as-a-Judge* (Zheng et al. 2023; Lifshitz, McIlraith, and Du 2025) 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 (Du et al. 2024; Chan et al. 2024; Chen et al. 2024b; Liang et al. 2023; Smit et al. 2024; Li et al. 2024c; Kim et al. 2024) 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 (Chen, Saha, and Bansal 2024; Wang et al. 2024a; Li et al. 2024a) refines samples by critically evaluating answers across different LLMs in iterative cycles. 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 (Shnitzer et al. 2023; Lu et al. 2023; Muqeeth et al. 2024; Ong et al. 2024), 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.

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. 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 more efficient and effective applications of LLMs.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 6250076060 and No. 62272219). This work was supported by the Shanghai Municipal Science and Technology Major Project.

References

Aggarwal, A. M. P.; Yang, Y.; and Mausam. 2023. Let's Sample Step by Step: Adaptive-Consistency for Efficient Reasoning and Coding with LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP*, 12375–12396.

Brown, B.; Juravsky, J.; Ehrlich, R.; Clark, R.; Le, Q. V.; Ré, C.; and Mirhoseini, A. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

Chan, C.; Chen, W.; Su, Y.; Yu, J.; Xue, W.; Zhang, S.; Fu, J.; and Liu, Z. 2024. ChatEval: Towards Better LLM-based Evaluators through Multi-Agent Debate. In *The Twelfth International Conference on Learning Representations, ICLR*.

Chen, J. C.; Saha, S.; and Bansal, M. 2024. ReConcile: Round-Table Conference Improves Reasoning via Consensus among Diverse LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL*, 7066–7085.

Chen, L.; Davis, J. Q.; Hanin, B.; Bailis, P.; Stoica, I.; Zaharia, M.; and Zou, J. 2024a. Are More LLM Calls All You Need? Towards Scaling Laws of Compound Inference Systems. *arXiv preprint arXiv:2403.02419*.

Chen, W.; Su, Y.; Zuo, J.; Yang, C.; Yuan, C.; Chan, C.; Yu, H.; Lu, Y.; Hung, Y.; Qian, C.; Qin, Y.; Cong, X.; Xie, R.; Liu, Z.; Sun, M.; and Zhou, J. 2024b. AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors. In *The Twelfth International Conference on Learning Representations, ICLR*.

Chen, X.; Aksitov, R.; Alon, U.; Ren, J.; Xiao, K.; Yin, P.; Prakash, S.; Sutton, C.; Wang, X.; and Zhou, D. 2023. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2311.17311*.

Christiano, P. F.; Leike, J.; Brown, T.; Martic, M.; Legg, S.; and Amodei, D. 2017. Deep reinforcement learning from human preferences. *Advances in Neural Information Processing Systems*, 30.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*.

Cui, G.; Yuan, L.; Wang, Z.; Wang, H.; Li, W.; He, B.; Fan, Y.; Yu, T.; Xu, Q.; Chen, W.; et al. 2025. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*.

Du, Y.; Li, S.; Torralba, A.; Tenenbaum, J. B.; and Mordatch, I. 2024. Improving Factuality and Reasoning in Language Models through Multiagent Debate. In *Forty-first International Conference on Machine Learning, ICML*.

Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T. B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; and Amodei, D. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Kim, K.; Lee, S.; Huang, K.-H.; Chan, H. P.; Li, M.; and Ji, H. 2024. Can llms produce faithful explanations for fact-checking? towards faithful explainable fact-checking via multi-agent debate. *arXiv preprint arXiv:2402.07401*.

Li, D.; Tan, Z.; Qian, P.; Li, Y.; Chaudhary, K. S.; Hu, L.; and Shen, J. 2024a. Smoa: Improving multi-agent large language models with sparse mixture-of-agents. *arXiv preprint arXiv:2411.03284*.

Li, J.; Zhang, Q.; Yu, Y.; Fu, Q.; and Ye, D. 2024b. More agents is all you need. *arXiv preprint arXiv:2402.05120*.

Li, Y.; Choi, D.; Chung, J.; Kushman, N.; Schrittwieser, J.; Leblond, R.; Eccles, T.; Keeling, J.; Gimeno, F.; Dal Lago, A.; et al. 2022. Competition-level code generation with alphacode. *Science*, 378(6624): 1092–1097.

Li, Y.; Du, Y.; Zhang, J.; Hou, L.; Grabowski, P.; Li, Y.; and Ie, E. 2024c. Improving multi-agent debate with sparse communication topology. *arXiv preprint arXiv:2406.11776*.

Li, Y.; Yuan, P.; Feng, S.; Pan, B.; Wang, X.; Sun, B.; Wang, H.; and Li, K. 2024d. Escape Sky-high Cost: Early-stopping Self-Consistency for Multi-step Reasoning. In *ICLR*.

Liang, T.; He, Z.; Jiao, W.; Wang, X.; Wang, Y.; Wang, R.; Yang, Y.; Shi, S.; and Tu, Z. 2023. Encouraging divergent thinking in large language models through multi-agent debate. *arXiv preprint arXiv:2305.19118*.

Lifshitz, S.; McIlraith, S. A.; and Du, Y. 2025. Multi-Agent Verification: Scaling Test-Time Compute with Multiple Verifiers. *arXiv preprint arXiv:2502.20379*.

Lightman, H.; Kosaraju, V.; Burda, Y.; Edwards, H.; Baker, B.; Lee, T.; Leike, J.; Schulman, J.; Sutskever, I.; and Cobbe, K. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR*.

Liu, H.; Zheng, Z.; Qiao, Y.; Duan, H.; Fei, Z.; Zhou, F.; Zhang, W.; Zhang, S.; Lin, D.; and Chen, K. 2024. MathBench: Evaluating the Theory and Application Proficiency of LLMs with a Hierarchical Mathematics Benchmark. In *Findings of the Association for Computational Linguistics: ACL 2024*, 6884–6915. Bangkok, Thailand: Association for Computational Linguistics.

Liu, R.; Gao, J.; Zhao, J.; Zhang, K.; Li, X.; Qi, B.; Ouyang, W.; and Zhou, B. 2025. Can 1B LLM Surpass 405B LLM? Rethinking Compute-Optimal Test-Time Scaling. *arXiv preprint arXiv:2502.06703*.

Lu, K.; Yuan, H.; Lin, R.; Lin, J.; Yuan, Z.; Zhou, C.; and Zhou, J. 2023. Routing to the expert: Efficient reward-guided ensemble of large language models. *arXiv preprint arXiv:2311.08692*.

Muqeeth, M.; Liu, H.; Liu, Y.; and Raffel, C. 2024. Learning to route among specialized experts for zero-shot generalization. *arXiv preprint arXiv:2402.05859*.

Ong, I.; Almahairi, A.; Wu, V.; Chiang, W.-L.; Wu, T.; Gonzalez, J. E.; Kadous, M. W.; and Stoica, I. 2024. RouteLLM: Learning to Route LLMs from Preference Data. In *The*

Thirteenth International Conference on Learning Representations.

Qian, C.; Xie, Z.; Wang, Y.; Liu, W.; Dang, Y.; Du, Z.; Chen, W.; Yang, C.; Liu, Z.; and Sun, M. 2024. Scaling large-language-model-based multi-agent collaboration. *arXiv preprint arXiv:2406.07155*.

Rae, J. W.; Borgeaud, S.; Cai, T.; Millican, K.; Hoffmann, J.; Song, F.; Aslanides, J.; Henderson, S.; Ring, R.; Young, S.; et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

Schick, T.; Dwivedi-Yu, J.; Dessì, R.; Raileanu, R.; Lomeli, M.; Hambro, E.; Zettlemoyer, L.; Cancedda, N.; and Scialom, T. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36: 68539–68551.

Shi, F.; Suzgun, M.; Freitag, M.; Wang, X.; Srivats, S.; Vosoughi, S.; Chung, H. W.; Tay, Y.; Ruder, S.; Zhou, D.; Das, D.; and Wei, J. 2023. Language models are multilingual chain-of-thought reasoners. In *The Eleventh International Conference on Learning Representations, ICLR*.

Shneiderman, B.; and Plaisant, C. 2010. *Designing the user interface: strategies for effective human-computer interaction*. ISBN 9780321537355.

Shnitzer, T.; Ou, A.; Silva, M.; Soule, K.; Sun, Y.; Solomon, J.; Thompson, N.; and Yurochkin, M. 2023. Large language model routing with benchmark datasets. *arXiv preprint arXiv:2309.15789*.

Smit, A. P.; Grinsztajn, N.; Duckworth, P.; Barrett, T. D.; and Pretorius, A. 2024. 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*, 45883–45905.

Snell, C.; Lee, J.; Xu, K.; and Kumar, A. 2024. Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters. *arXiv preprint arXiv:2408.03314*.

Stiennon, N.; Ouyang, L.; Wu, J.; Ziegler, D.; Lowe, R.; Voss, C.; Radford, A.; Amodei, D.; and Christiano, P. F. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33: 3008–3021.

Talebirad, Y.; and Nadiri, A. 2023. Multi-agent collaboration: Harnessing the power of intelligent llm agents. *arXiv preprint arXiv:2306.03314*.

Taubenfeld, A.; Sheffer, T.; Ofek, E.; Feder, A.; Goldstein, A.; Gekhman, Z.; and Yona, G. 2025. Confidence improves self-consistency in llms. *ACL*.

Wan, Z.; Li, Y.; Song, Y.; Wang, H.; Yang, L.; Schmidt, M.; Wang, J.; Zhang, W.; Hu, S.; and Wen, Y. 2025. ReMA: Learning to Meta-think for LLMs with Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:2503.09501*.

Wang, J.; Wang, J.; Athiwaratkun, B.; Zhang, C.; and Zou, J. 2024a. Mixture-of-Agents Enhances Large Language Model Capabilities. *arXiv preprint arXiv:2406.04692*.

Wang, Q.; Wang, Z.; Su, Y.; Tong, H.; and Song, Y. 2024b. Rethinking the Bounds of LLM Reasoning: Are Multi-Agent Discussions the Key? In *62nd Annual Meeting of the Association for Computational Linguistics, ACL 2024*, 6106–6131. Association for Computational Linguistics (ACL).

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q. V.; Chi, E. H.; Narang, S.; Chowdhery, A.; and Zhou, D. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations, ICLR*.

Wang, Y.; Ma, X.; Zhang, G.; Ni, Y.; Chandra, A.; Guo, S.; Ren, W.; Arulraj, A.; He, X.; Jiang, Z.; et al. 2024c. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35: 24824–24837.

Welleck, S.; Liu, J.; Lu, X.; Hajishirzi, H.; and Choi, Y. 2022. Naturalprover: Grounded mathematical proof generation with language models. *Advances in Neural Information Processing Systems*, 35: 4913–4927.

Yadkori, Y. A.; Kuzborskij, I.; György, A.; and Szepesvári, C. 2024. To Believe or Not to Believe Your LLM: Iterative Prompting for Estimating Epistemic Uncertainty. In Globerson, A.; Mackey, L.; Belgrave, D.; Fan, A.; Paquet, U.; Tomczak, J.; and Zhang, C., eds., *Advances in Neural Information Processing Systems*, volume 37, 58077–58117. Curran Associates, Inc.

Yang, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Li, C.; Liu, D.; Huang, F.; Wei, H.; et al. 2024. Qwen2. 5 Technical Report. *arXiv preprint arXiv:2412.15115*.

Zhang, H.; Cui, Z.; Wang, X.; Zhang, Q.; Wang, Z.; Wu, D.; and Hu, S. 2025a. If Multi-Agent Debate is the Answer, What is the Question? *arXiv preprint arXiv:2502.08788*.

Zhang, J.; Xu, X.; Zhang, N.; Liu, R.; Hooi, B.; and Deng, S. 2024a. 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)*, 14544–14607.

Zhang, L.; Hosseini, A.; Bansal, H.; Kazemi, M.; Kumar, A.; and Agarwal, R. 2024b. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*.

Zhang, Y.; Ye, P.; Yang, X.; Feng, S.; Zhang, S.; Bai, L.; Ouyang, W.; and Hu, S. 2025b. Nature-inspired population-based evolution of large language models. *arXiv preprint arXiv:2503.01155*.

Zheng, L.; Chiang, W.-L.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, Z.; Li, D.; Xing, E.; et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36: 46595–46623.