Majority of the Bests: Improving Best-of-N via Bootstrapping

Anonymous Authors[1]

Abstract

Inference-time computational methods significantly enhance the reasoning abilities of Large Language Models (LLMs). Among these, Best-of-N has gained attention for its simplicity and scalability. It generates N solutions from the LLM and selects the best one based on the reward model's evaluation. Due to imperfect rewards, even with a large N, the probability of selecting the correct answer does not necessarily converge to one. To mitigate this limitation, we propose Majority-of-the-Bests (MoB), a novel and hyperparameter-free selection mechanism that estimates the output distribution of Best-of-N via bootstrapping and selects its mode. Experimental results across five benchmarks, three different base LLMs, and two reward models demonstrate consistent improvements over Best-of-N in 25 out of 30 setups. We further provide theoretical results for the consistency of the bootstrapping.

1. Introduction

Scaling the inference-time computation of language models (LMs) has led to a significant improvement of their performance on a variety of tasks (Brown et al., 2024; Snell et al., 2024; Wu et al., 2024; OpenAI, 2024; DeepSeek-AI, 2025). A growing number of methods have been introduced in this paradigm, such as generating long chains-of-though (Wei et al., 2022; Muennighoff et al., 2025), asking the model to evaluate and improve its own outputs (Madaan et al., 2023), and tree search (Yao et al., 2023; Hao et al., 2023; Zhang et al.). Another family of such algorithms, termed *sample-and-marginalize* by Wang et al. (2022), generate multiple outputs from the model and then aggregate them into a final answer. Examples include Self-consistency (Wang et al., 2022), Best-of-N (Lightman et al., 2023), and Weighted Best-of-N (Li et al., 2022). These methods have gained popularity due to their simplicity and scalability.

Self-consistency (SC) (Wang et al., 2022), also referred to as "majority voting", is a widely used algorithm in sample-and-marginalize. It samples multiple outputs from the model and selects the final answer that appears most frequently among them. SC improves the performance by leveraging a key property of the model's output distribution: on difficult problems, the probability of generating the correct answer is often far from 1, making single-sample predictions unreliable. SC capitalizes on the fact that, even if the model's output distribution is imperfect, it may still favor the correct answer and generate it more frequently than incorrect ones.

Best-of-N (BoN) (Lightman et al., 2023) uses a reward model to evaluate the generated outputs and chooses the final answer in the highest-scoring output. With an ideal reward model, BoN succeeds as long as one of the generated outputs is correct. In this paper, we highlight that in the realistic setting of an imperfect reward model, the success of BoN is no longer (nearly) guaranteed. In such cases, BoN exhibits stochastic behavior akin to the underlying generative model. While the reward model improves the likelihood of selecting the correct answer, it often falls short of ensuring certainty. This is the same property that underlies the effectiveness of SC. Motivated by this observation, we show that applying a similar principle—aggregating multiple samples to identify the most probable answer—leads to a better performance over BoN.

We introduce Majority-of-the-Bests (MoB), a method that leverages bootstrapping to improve upon BoN *by approximating the most probable output of BoN*. As illustrated in Figure 1, after obtaining multiple (parallel) solution samples for a given question and computing their rewards, we apply bootstrapping: we create subsets of size m by sampling with replacement from the generated outputs. For each subset, we select the sample with the highest reward. This results in a new set of high-reward samples, over which we perform majority voting to determine the final answer. Just like BoN and SC, MoB can be applied independent of the output generation procedure. It only modifies the selection of the final answer with marginally extra computation on the CPU. We provide a procedure to adaptively select m and eliminating any critical hyperparameters from the algorithm. We show the consistency of the algorithm theoretically, and

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

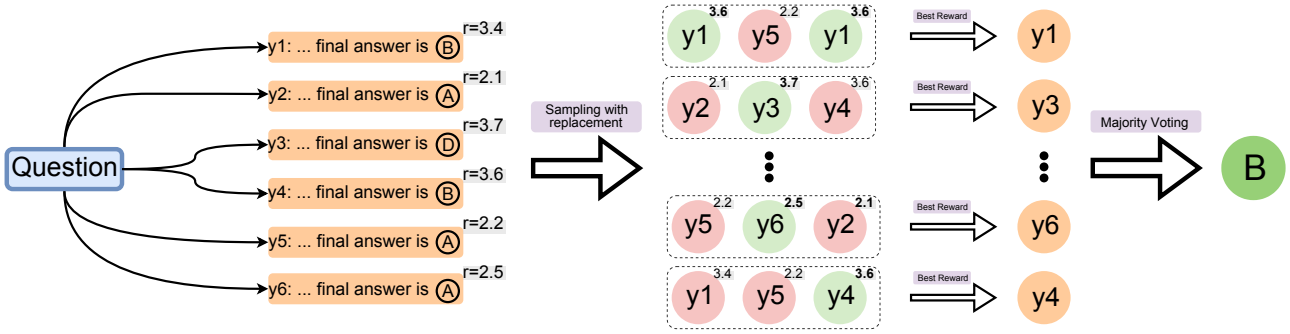

Figure 1: Majority-of-the-Bests: first, N outputs are generated for the given question. Then, we create a large number of subsets of size $m < N$ by sampling with replacements from the generated outputs. From each subset, we choose the output with the highest reward. The most frequent answer among these chosen outputs is reported as the final answer.

empirically show significant improvements over BoN on 25 out of 30 tested setups.

2. Background

In this section, we formulate BoN and bootstrapping and provide some background for the algorithm and its theoretical grounds. Given a prompt x, in the standard procedure with LLMs, we sample an output $Y \sim p_{\text{ref}}$ from a base model p_{ref}. This output yields a corresponding final answer $Z = f(Y)$ after applying a post-processing or evaluation function f. For example, for a multiple choice problem, Z is the chosen option and Y is the whole output containing both Z and its justification. We denote the distribution of the final answer in this procedure as π_{ref}, that is, $Z \sim \pi_{\text{ref}}$. The goal is to find the correct final answer z^*. We define the success probability for this given problem as the probability of selecting the correct final answer. If the algorithm's final answer is Z, the success probability is defined as $\mathbb{P}(Z = z^*)$. Given a dataset of questions, the average of the success probabilities over all questions is referred to as the accuracy. For the standard procedure, the success probability is equal to $\pi_{\text{ref}}(z^*)$ and the corresponding accuracy is called the *pass@1* accuracy. We assume access to a reward model r that assigns a reward $R = r(Y)$ to the output Y, reflecting its accuracy, coherence, or alignment with human preferences (Uesato et al., 2022; Lightman et al., 2023).

Reward models can be categorized into two distinct groups: Outcome-supervised Reward Models (ORMs) and Process-based Reward Models (PRMs). ORMs are trained to predict a scalar reward by evaluating the quality or correctness of the generated output. In contrast, PRMs are trained to provide rewards for each intermediate step or component within a generation process, thereby offering more granular feedback on the reasoning or constructive path taken to reach an outcome (Uesato et al., 2022; Lightman et al., 2023).

For a given budget N, sample-and-marginalize algorithms generate N independent outputs $Y_1, \ldots, Y_N \sim p_{\text{ref}}$ and select the final answer reached by one of these outputs. BoN selects the final answer from the output with the highest reward, that is,

$$Z_N^{\text{Best}} = f\left(\underset{y \in \{Y_1, \ldots, Y_N\}}{\text{argmax}} \; r(y) \right).$$

Alternatively, self-consistency or majority voting selects the final answer that occurs most frequently among $Z_1, \ldots, Z_N$ where $Z_i = f(Y_i)$ is the final answer for output Y_i. If N is large enough, this most frequent answer will be the mode of the final answer distribution π_{ref}. Li et al. (2022) suggested the Weighted Best-of-N (WBoN) selection method. For each final answer, WBoN sums the rewards of all outputs that lead to it. Then it selects the final answer with the highest cumulated reward.

Bootstrapping. Bootstrapping is a powerful and widely used non-parametric resampling technique for estimating the distribution of a statistic by repeatedly drawing samples with replacement from the original dataset (Efron, 1992; Efron and Tibshirani, 1994). The core idea is to generate multiple "bootstrap samples", by sampling observations uniformly and with replacement. For each bootstrap sample, the statistic of interest is computed. The collection of these computed statistics from the many bootstrap samples forms an empirical approximation of the statistic's true distribution. We use this technique to approximate the distribution of BoN's output.

3. Motivation: Output Distribution of Best-of-N

To motivate our algorithm, we highlight the behavior of BoN's final answer distribution. We denote this distribution

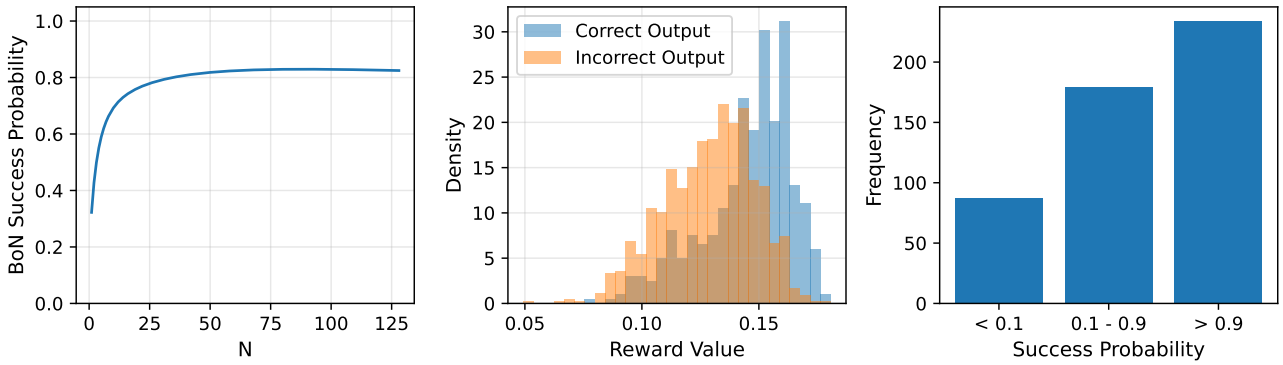

Figure 2: *(Left)* BoN's success probability as a function of N for question 647 from MMLU-Pro-Math. The success probability remains bellow 80%. *(Middle)* Distribution of the reward for correct and incorrect outputs for the same question. A separation between the two distributions is ideal. *(Right)* Histogram of Best-of-64 success probabilities over 500 questions.

by π_N. It means,

$$Z_N^{\text{Best}} \sim \pi_N.$$

Assume among the N sampled outputs, N_c outputs $\{Y_1^c, \dots, Y_{N_c}^c\} \subseteq \{Y_i\}_{i=1}^N$ yield the correct final answer: $f(Y_i^c) = z^*$. Conversely, $N_w = N - N_c$ outputs $\{Y_1^w, \dots, Y_{N_w}^w\} \subseteq \{Y_i\}_{i=1}^N$ lead to an incorrect solution. Then, BoN's output is correct if the highest reward among the correct outputs is larger than the highest reward among the incorrect ones. Formally, we can express this condition as:

$$\max\big(r(Y_1^c), \dots, r(Y_{N_c}^c)\big) > \max\big(r(Y_1^w), \dots, r(Y_{N_w}^w)\big). \tag{1}$$

There are two factors that influence the probability of this event. First, note that each side of (1) is the maximum of some random variables. As the number of random variables increases, the probability distribution of their maximum shifts towards higher values. Therefore, larger values of N_c and smaller values of N_w, make condition (1) more likely. The values of N_c and N_w depend on $\pi_{\text{ref}}(z^*)$, the probability of the correct answer z^* in the base model's final answer distribution π_{ref}. For large enough n, we will have

$$N_c \approx N \cdot \pi_{\text{ref}}(z^*) \quad , \quad N_w \approx N \cdot (1 - \pi_{\text{ref}}(z^*)).$$

It means that if the base model has a higher chance of solving the problem, BoN is also more likely to select the correct answer.

The second factor is the distribution of $r(Y_i^c)$ and $r(Y_i^w)$ on each side of (1). The reward of a correct output follows the conditional distribution $\mathcal{P}_c \triangleq \mathbb{P}(r(Y)|f(Y) = z^*)$ while the reward of an incorrect output follows the conditional distribution $\mathcal{P}_w \triangleq \mathbb{P}(r(Y)|f(Y) \neq z^*)$. We hope that the reward model assigns higher rewards to correct outputs, and $r(Y_i^c) \sim \mathcal{P}_c$ on the left side of (1) generally be larger than $r(Y_i^w) \sim \mathcal{P}_w$ on the right side.

Therefore, the success probability of BoN heavily depends on the separation between $\mathcal{P}_c$ and $\mathcal{P}_w$. A perfect reward model would always assign a higher value to a correct output than to an incorrect one. In that case, as long as at least one correct output is generated (which is highly likely for large enough N), condition (1) is satisfied. The resulting success probability is close to 1, indicating a nearly deterministic final answer. On the other hand, consider the case where $\mathcal{P}_c$ and $\mathcal{P}_w$ are identical. In this case, the reward of an output becomes independent of its correctness, and choosing according to the reward model will be no better than a random choice. Consequently, the success probability of BoN will be the same as the base model, i.e. $\pi_N(z^*) = \pi_{\text{ref}}(z^*)$. We provide a complete theoretical analysis in Appendix A.1. In practice, our reward models exhibit a middle ground between these two extremes. They might not be perfect for BoN to succeed with a single correct output, but they can still be somewhat informative to increase the success probability of BoN compared to the base model.

In Figure 2, we show an example of these dynamics for Question 647 of the MMLU-Pro-Math benchmark (Wang et al., 2024b) with base model `Qwen2.5-3b-instruct` (Qwen Team, 2024) and reward model `ArmoRM` (Wang et al., 2024a). We approximate the output distribution p_{ref} with a large pool of 1400 samples. In Question 647 (Figure 2), the two distributions $\mathcal{P}_c$ and $\mathcal{P}_w$ are overlapping, and even with large values of N, the success probability remains below 80%. Nonetheless, BoN still outperforms the base model, which is equivalent to Best-of-1 and has a success probability of 30% in this case.

We expect the stochasticity of BoN's output to depend on the difficulty of the question relative to the base and re-

ward models' capabilities. For more difficult questions, the base model generates fewer correct outputs, and the reward model is less likely to distinguish the correct outputs from the incorrect ones. Through the two factors discussed above, BoN is not able to pick the correct answer with high certainty. The right plot in Figure 2 shows the histogram of the success probability of Best-of-64 among 500 randomly selected MMLU-Pro-Math problems. We see that for approximately 175 problems, BoN has a success probability between 0.1 and 0.9. That means, BoN has a significant chance of returning the correct answer but fails to do so reliably. *The idea behind our introduced method, MoB, is that if we can find the most probable output of the BoN distribution, we may reliably pick the correct answer even if its probability is well below* 1.

4. Majority-of-the-Bests

In Section 3, we showed that BoN's final answer is stochastic, and this stochasticity might remain true even with a very large budget N. In this section, we introduce Majority-of-the-Bests (MoB). MoB can select the correct answer with high probability as long as the correct answer is the most probable output of BoN, even if its probability is well below 1. We first showcase this idea in the hypothetical case where BoN's output distribution π_N is given by an oracle. Later, we show how to estimate this distribution using bootstrapping.

4.1. MoB with Oracle Access to BoN's Output Distribution

Suppose the distribution of BoN's final answer π_N is known through an oracle. Instead of sampling from this distribution, which is equivalent to BoN and is a noisy decision, we propose selecting the mode of this distribution. That is

$$z_N^{\text{OracleMoB}} = \operatorname*{argmax}_z \pi_N(z). \tag{1}$$

We refer to this algorithm as *Oracle MoB* as it relies on an oracle. By selecting the mode, if the correct answer has a higher probability than any of the other answers, it will be selected without any randomness that would reduce the success probability. Since $\pi_{\text{ref}} = \pi_1$, we can say SC for a large N is equivalent to Oracle MoB with $N = 1$. It has been extensively shown that SC improves the LLM's original accuracy. As we will also empirically show, MoB similarly increases the accuracy of BoN by selecting the mode of its output distribution.

In Figure 3, we compare the accuracy of Oracle MoB with BoN on MATH500 (Lightman et al., 2023; Hendrycks et al., 2021) and math problems of MMLU-Pro (Wang et al., 2024b). We use the same output pool, base model, and reward model as Figure 2. We can see that depending on the value of N, Oracle MoB provides 5 to 10 percent points improvement in accuracy. Oracle MoB unrealistically requires an oracle access to π_N. Next, we will show how π_N can be estimated via bootstrapping and remove the oracle dependence.

4.2. MoB with Estimated BoN's Output Distribution

We now discuss how, without the oracle access to the BoN's output distribution π_N, one can approximately find its most probable output. The most obvious approach is to follow the same procedure as SC. For some $k \geq 1$, we can run k independent BoN procedures. Then, out of the k resulting answers, we select the final answer that appears the most number of times. The answer of the BoN procedures let us approximate π_m, and selecting the most frequent answer among them will approximate Oracle MoB (1) with budget m. We refer to this algorithm as "BoN+SC" due to its simple combination of BoN and SC. To keep the generation budget fixed at N, we are forced to use a smaller budget m for each of the BoN runs. For now, we treat the choice of m as a hyperparameter, but will return to this choice later. Assume $m < N$ and $k = \lfloor N/m \rfloor$. Formally,

$$Z_m^{\text{Best},(i)} = f\left(\operatorname*{argmax}_{y \in \{Y_{im}, \ldots, Y_{(i+1)m-1}\}} r(y)\right)(i = 1, \ldots, k), \tag{2}$$

$$Z_{m,n}^{\text{BoN + SC}} = \operatorname*{argmax}_z \sum_i \mathbb{I}\left[Z_m^{\text{Best},(i)} = z\right]. \tag{3}$$

The main problem with BoN+SC is that it is too expensive. We would like to have a large value for m to get the benefits offered by BoN, and to have a fairly accurate estimation of π_m, we need a reasonably large value for k. Together, this requires a large budget $N \approx mk$.

The deficiency of BoN+SC comes from the fact that each sample Y_i only contributes to generating one BoN output. To address this deficiency, we propose estimating π_m not by generating independent samples from it, but by bootstrapping. To do that, we first note that the distribution π_m of Z_m^{Best} is a function of the unknown distribution p_{ref}. Bootstrapping suggests to estimate π_m with the BoN's output distribution under a known approximation $\hat{p}_{\text{ref}} \approx p_{\text{ref}}$. The typical non-parametric approach is to set $\hat{p}_{\text{ref}}$ to be the empirical distribution of the generated samples $\{Y_1, \ldots, Y_N\}$. Since $\hat{p}_{\text{ref}}$ is known, we can cheaply sample from it. For any arbitrarily large value B, we generate B approximately sampled BoN outputs. We first create B datasets of size m from $\hat{p}_{\text{ref}}$. That is

$$D_i = \{\hat{Y}_{i,1}, \hat{Y}_{i,2}, \ldots, \hat{Y}_{i,m}\} \sim \hat{p}_{\text{ref}}, \quad (i = 1, \ldots, B).$$

This is equivalent to sampling m outputs from the original pool $\{Y_1, \ldots, Y_n\}$ with replacement. Then, similar to

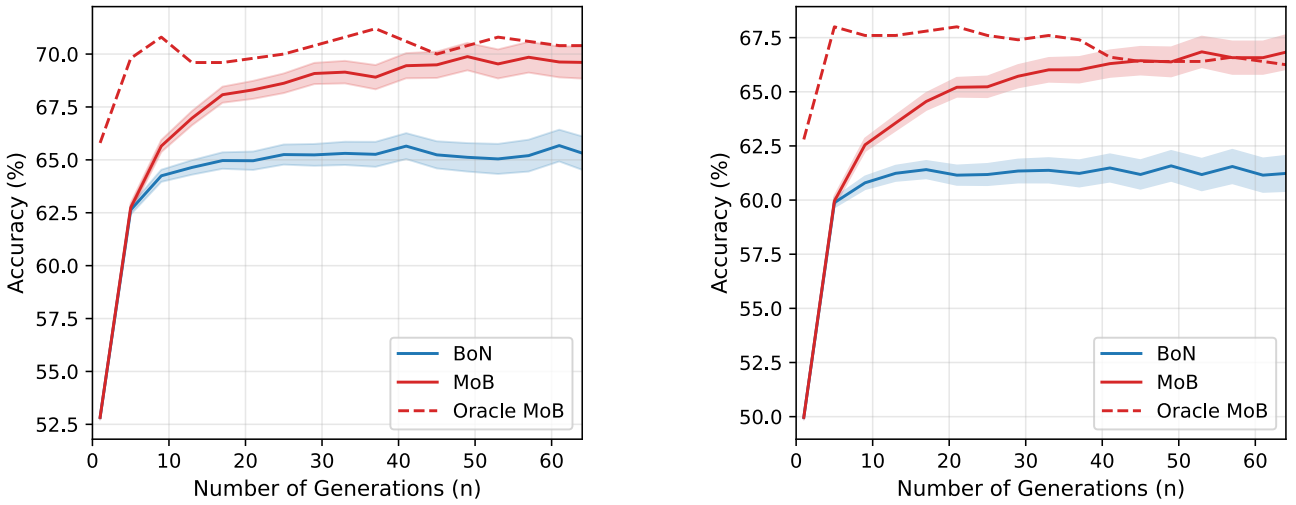

Figure 3: Final answer accuracy comparison of BoN, MoB, and Oracle MoB on MMLU-Pro-Math using `Qwen2.5-3b-instruct` *(Left)* and `Llama3.1-8b-instruct` *(Right)* as the base model, and `ArmoRM` as the reward model. Results are averaged across all problems and multiple runs. Shaded area indicates the standard error.

BoN+SC, we can run BoN on each dataset, and then pick the most common outcome. Formally,

$$\hat{Z}_m^{\text{Best},(i)} = f\left(\operatorname*{argmax}_{y \in D_i} r(y)\right) \qquad (i = 1, \ldots, B), \tag{4}$$

$$Z_{m,N}^{\text{MoB}} = \operatorname*{argmax}_z \sum_{i=1}^{B} \mathbb{I}\left[\hat{Z}_m^{\text{Best},(i)} = z\right]. \tag{5}$$

This procedure is our MoB algorithm for a given m. We define $\hat{\pi}_{m,N}$ to be the (random) distribution of $\hat{Z}_m^{\text{Best},(1)}$ given $\{Y_i\}$ at hand. With sufficiently large B (usually $B = 10,000$ is sufficient), the empirical distribution of $\{\hat{Z}_m^{\text{Best},(i)}\}$ will accurately estimate $\hat{\pi}_{m,N}$. With this approximation, we can write

$$Z_{m,N}^{\text{MoB}} \approx \operatorname*{argmax}_z \hat{\pi}_{m,N}(z) \tag{6}$$

Note that this is a light computation that can be carried out on the CPU. Therefore, we can freely choose a large B. In the supplementary material, we provide an even more efficient way of estimating $\hat{\pi}_{m,N}$ with $\mathcal{O}(N \log N)$ complexity.

In Figure 4, we compare MoB with BoN+SC in the same setup as Figure 3. In the left plot, we fix $m = 8$ and compare the algorithms' error on estimating π_m for a range of values for N. We measure the distance between the two distributions according to the ℓ_1-norm. As we can see, bootstrapping is consistently the superior approach for this approximation task and offers a more accurate estimation of π_m. In the right plot, we set $m = \lfloor\sqrt{N}\rfloor$ and compare the final accuracy of the algorithms. The choice of $m = \lfloor\sqrt{N}\rfloor$ ensures that $k \approx \sqrt{N}$ and will also increase as N increases. We observe that the superior accuracy of bootstrapping in the estimation of π_m translates to a better final accuracy of the algorithm, especially when the budget N is more limited.

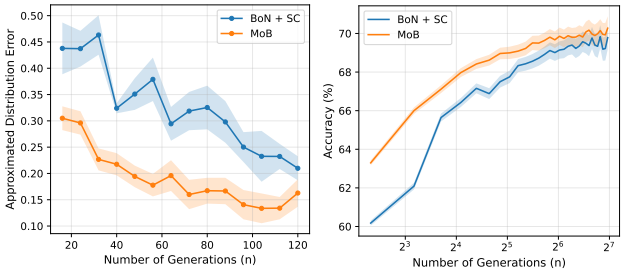

Figure 4: Comparison of MoB and BoN+SC using `Qwen2.5-3b-instruct` (reference model) with `ArmoRM` as the reward model. Left: ℓ_1 error of π_m for m=8. Right: average accuracy on MMLU-Pro-Math. Shaded areas show standard error.

One might wonder if it is possible to choose m to be much larger than what was possible in BoN+SC, potentially even $m = N$. There is no obvious limitation on the size of resampled datasets D_i, and nonetheless, most commonly in bootstrapping, the size of resampled datasets is equal to the original dataset. However, estimating the distribution of values related to the extremes of random samples is a classic example of failure for the conventional bootstrapping, see for example Athreya and Fukuchi (1994) and Efron and Tibshirani (1994, Section 7.4). Since BoN selects the output with the highest reward, it is affected by the same failure. To see this, note that the output with the highest reward appears in each dataset with the probability of $1 - (\frac{N-1}{N})^m$, and it

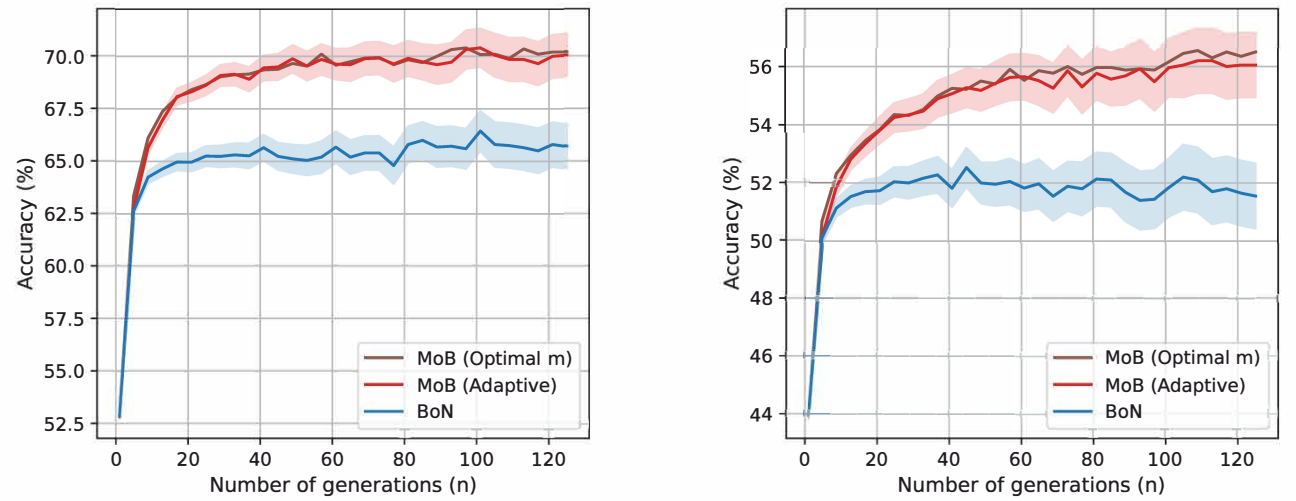

Figure 5: Average answer accuracy comparison using `ArmoRM` reward model with MMLU-Pro-Math and `Qwen2.5-3b-instruct` *(Left)* and MATH500 and `Llama3.1-8b-instruct` *(Right)*. Shaded area indicates the standard error.

will be chosen in any dataset in which it appears. Therefore, if $m = N$,

$$\mathbb{P}\Big(\hat{Z}_m^{\text{Best},(i)} = Z_N^{\text{Best}}\Big) \geq 1-\big(\frac{N-1}{N}\big)^N \approx 1-e^{-1} \approx 0.632.$$

This means that $\hat{\pi}_{N,N}$ will always assign a probability of at least 0.632 to the conventional BoN's answer. As we discussed in Section 3, π_N might be quite stochastic, which means such approximation cannot be accurate. Even more critically for our use of this approximation, the mode of $\hat{\pi}_{N,N}$ will always coincide with BoN's answer, and MoB becomes equivalent to BoN.

Fortunately, using smaller resampled datasets, as we do in MoB, is one of the remedies for such failures of bootstrapping and is well-studied in the literature, (Athreya and Fukuchi, 1994; Bickel et al., 2011) and is referred to as m-out-of-n bootstrapping. We show that under the usual conditions of m-out-of-n bootstrapping and mild assumptions on the tail of reward distributions, our use of bootstrapping to estimate π_m is a valid one. Similar to the typical guarantees for bootstrap estimations, we show that our bootstrap estimation is indeed consistent.

Theorem 4.1. *Under mild assumptions on the tail of distribution of rewards, if there are finite possible values for Z and as $N \to \infty$, we have $m \to \infty$ and $m/N \to 0$, then for any $\epsilon > 0$, the estimated $\hat{\pi}_{m,N}$ will converge to the true distribution π_m. That is,*

$$\lim_{n\to\infty} \mathbb{P}\big(\|\hat{\pi}_{m,N} - \pi_m\|_1 \geq \epsilon\big) = 0.$$

We defer the exact technical statement and proof to the supplementary material. Theorem 4.1 shows that the estimated distribution $\hat{\pi}_{m,N}$ will match the true BoN output distribution π_m. It means that MoB with bootstrapped distribution in (6) will reach the same accuracy as its oracle version in (1), but with a larger required budget due to $m < N$. To achieve this, it suffices to pick m such that the condition of Theorem 4.1 holds, which is possible by simply using a fix schedule of the form $m(n) = n^\alpha$ for some $0 < \alpha < 1$. In the next section, we will discuss the choice of m in more detail and provide a procedure to choose m automatically.

4.3. Adaptive Subsample Size m

The choice of m imposes a trade-off. A larger value of m means that we are running BoN with a larger number of samples. Since we expect the success probability of BoN to increase with more samples, this means that the mode of π_m will be more likely to be correct. On the other hand, as m becomes larger and closer to n, our estimate $\hat{\pi}_{m,N}$ of π_m becomes more inaccurate. As we saw in Section 4.2, bootstrapping might fail to provide a consistent estimate if $m = N$.

Ideally, we would like to find an m such that our final answer $Z_{m,N}^{\text{MoB}}$ based on the estimated distribution as in (6) becomes closest to the Oracle MoB (1) of Section 4.1. The natural approach for this goal is to find the value of m that minimizes the distance between $\hat{\pi}_{m,N}$ and π_N, that is

$$M_N^* = \operatorname*{argmin}_m \|\hat{\pi}_{m,N} - \pi_N\|_1. \tag{7}$$

This minimization problem automatically captures both aspects of the trade-off. Large values of m make π_m, which is approximated by $\hat{\pi}_{m,N}$ closer to π_N, but at the same time

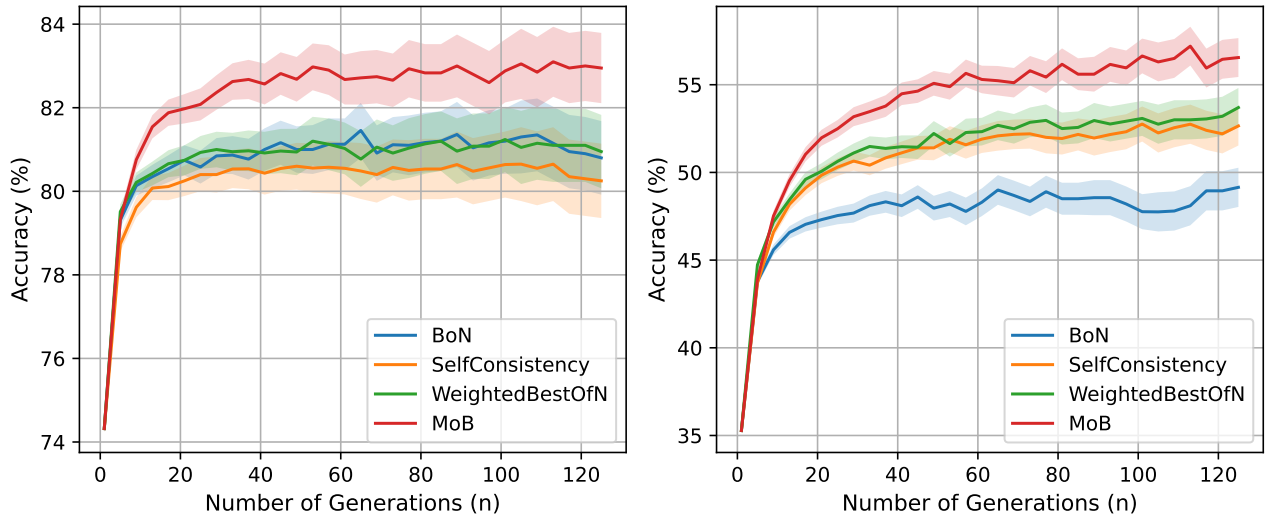

Figure 6: Accuracy comparison on different datasets using base model `Qwen2.5-3b-instruct` and `GRM` reward model. Standard deviation is shown as the shaded area. *(Left)*: GSM8k. *(Right)*: MMLU-Pro-Chem.

if m is too large, the error of this approximation becomes too large and increase the objective $\|\hat{\pi}_{m,N} - \pi_N\|_1$.

Unfortunately, the distribution π_N in the objective of (7) is unknown, and therefore cannot be used in practice. The theoretical results by Götze and Račkauskas (2001) show that if Z only takes two possible values and under some other technical conditions, the distance $\|\hat{\pi}_{m,N} - \hat{\pi}_{m/2,N}\|_1$ is proportional to the one in (7)[1]:

$$\|\hat{\pi}_{m,N} - \hat{\pi}_{m/2,N}\|_1 \propto \|\hat{\pi}_{m,N} - \pi_N\|_1.$$

Inspired by this result, Bickel and Sakov (2008) provides some optimality results for choosing m by minimizing the more general loss $\|\hat{\pi}_{m,N} - \hat{\pi}_{qm,N}\|_1$ for some $0 < q < 1$ instead of just $q = 0.5$ considered by Götze and Račkauskas (2001).

Based on the findings of Bickel and Sakov (2008), we propose using the following approach to pick m. We first consider the candidates of the form $\lfloor q^j N \rfloor$ and pick the value among them that minimizes $\|\hat{\pi}_{m,N} - \hat{\pi}_{qm,N}\|_1$.

$$m_j = \lfloor q^j n \rfloor \qquad (j = 0, 1, 2, \ldots),$$
$$\hat{M}^*_N = \operatorname*{argmin}_{m=m_j} \|\hat{\pi}_{m_j,N} - \hat{\pi}_{m_{j-1},N}\|_1.$$

Note that this involves calculating the approximating $\hat{\pi}_{m,N}$ for all values of m_j. These will be just $\mathcal{O}(\log n)$ distributions and computationally cheap. Finally, MoB's output is

$$Z^{\text{MoB}}_N = Z^{\text{MoB}}_{\hat{M}^*_N,N}. \tag{8}$$

The choice of q has been observed not to be critical in most applications. Bickel and Sakov (2008) observes no significant difference among $q = 0.75, 0.65, 0.6, 0.5$. In our experiments, we fix $q = 0.75$. In Figure 5, we evaluate the efficiency of this procedure to select m. For each N, we find the optimal m by evaluating the accuracy of the resulting MoB output for a set of candidate values. Specifically, we choose from $\{N^\alpha\}$ for $\alpha \in [0.1, 0.9]$. We call the accuracy of this optimal m, MOB (optimal m). We plot the accuracy of our adaptive m approach against this for two different settings. These figures show that adaptive m performance closely follows the optimal m variant.

5. Experiments

We conducted a series of experiments to compare the performance of our proposed method against several well-known test-time sample-and-marginalize approaches across a range of datasets, generative models, and reward models. The datasets include MATH500 (Lightman et al., 2023), GSM8K (Cobbe et al., 2021b), MMLU-Pro-Math and Chem (Wang et al., 2024b), and CommonSenseQA (Talmor et al., 2018). We have experimented with three different generative models from different families and different sizes: Qwen2.5-3B-Instruct (Qwen Team, 2024), Llama-3.1-8B-Instruct (Grattafiori et al., 2024), and Gemma-2-9B (Team et al., 2024). For reward models, we used two widely adopted

[1]This is a rough interpretation of the results by Götze and Račkauskas (2001), where the ratio of the two losses is studied. We refer the reader to the original paper for more details.

Table 1: Qwen2.5-3B and GRM3B as base and reward models, $N = 128$

	MATH500	MMLU-PRO-MATH	MMLU-PRO-CHEM	GSM8K	COMMONSENSEQA
BON	64.15±1.07	66.00±1.06	49.15±1.12	80.80±0.88	77.70±0.93
SC	66.30±1.06	65.70±1.06	52.65±1.12	80.25±0.89	76.15±0.95
WBON	67.30±1.05	64.55±1.07	53.70±1.12	80.95±0.88	54.75±1.11
MOB (OURS)	**69.80**±1.03	**69.65**±1.03	**56.55**±1.11	**82.95**±0.84	**77.40**±0.94

Table 2: Results on MATH500 across all base and reward models (N=128)

	ARMORM			GRM		
	GEMMA-2-9B	LLAMA3.1-8B	QWEN2.5-3B	GEMMA-2-9B	LLAMA3.1-8B	QWEN2.5-3B
BON	52.20±1.12	51.65±1.12	60.50±1.09	53.85±1.11	56.55±1.11	64.15±1.07
SC	52.65±1.12	61.15±1.09	66.30±1.06	52.65±1.12	61.15±1.09	66.30±1.06
WBON	53.60±1.12	**63.10**±1.08	67.05±1.05	56.00±1.11	**63.65**±1.08	67.30±1.05
MOB (OURS)	**56.65**±1.11	**62.50**±1.08	**68.25**±1.04	**57.80**±1.10	**64.15**±1.07	**69.80**±1.03
↑MOB OVER BON	4.45±1.57	10.85±1.56	7.75±1.51	3.95±1.57	7.60±1.54	5.65±1.48

ORMs: ArmoRM (Wang et al., 2024a) and GRM (Yang et al., 2024), with 8B and 3B parameters, respectively. These choices result in thirty diverse experimental setups that rigorously evaluate our method's performance.

Figure 6 presents the accuracy of different methods on GSM8K and MMLU-Pro-Chem across varying values of N. Our method consistently outperforms the baselines, showing clear improvements even at smaller N values. Table 1 presents the accuracy of our method alongside SC, BoN, and WBoN for $N = 128$ across all benchmarks, using `Qwen2.5-3b-instruct` and `GRM` as the base and reward models, respectively. Our method achieves state-of-the-art performance on all five benchmarks. In Table 2, we report the accuracy on MATH500 for all base and reward model combinations. This table also includes a row showing the performance improvement of our method over BoN. As shown in Table 2, MoB consistently outperforms BoN in every setting. These results show the potential of MoB as a drop-in replacement of BoN in all these widely used experimental setups with negligible additional CPU computation and no extra hyperparameters. Complete results for all thirty experiment configurations are provided in the Appendix.

6. Conclusion and Future Work

In this paper, we examined how imperfect reward models can lead to distributional overlap between correct and incorrect answers in Best-of-N (BoN), often resulting in incorrect selections. To address this, we introduced Majority-of-the-Bests (MoB), a bootstrapped method designed to improve estimation of the BoN distribution. MoB achieves superior performance compared to other selection algorithms, BoN, self-consistency (SC), and WeightedBoN (WBon) outperforming them in 25 out of 30 experimental setups. Our method is scalable, requires no hyperparameter tuning, and adds only negligible CPU computational overhead. We propose that MoB can serve as a drop-in replacement for BoN in any task that involves selecting a final answer based on noisy reward signals. Looking forward, we believe MoB's selection signal could enable early stopping in parallel LLM generation, or be applied more broadly in any framework that relies on sampling from an LLM. However, MoB is limited to settings where the task requires producing a final answer, and like all sampling-based methods, it incurs higher inference costs compared to zero-shot approaches.

References

K. B. Athreya and J. Fukuchi. Bootstrapping extremes of I.I.D. random variables. In *Proceedings of the Conference on Extreme Value Theory and Applications, Volume 3*. National Institute of Standards and Technology (NIST), 1994.

Peter J Bickel and Anat Sakov. On the choice of m in the m out of n bootstrap and confidence bounds for extrema. *Statistica Sinica*, pages 967–985, 2008.

Peter J Bickel, Friedrich Götze, and Willem R van Zwet. Resampling fewer than n observations: gains, losses, and remedies for losses. In *Selected works of Willem van Zwet*, pages 267–297. Springer, 2011.

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.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems, 2021. *URL https://arxiv. org/abs/2110.14168*, 9, 2021b.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. https://github.com/deepseek-ai/DeepSeek-R1/blob/main/DeepSeek_R1.pdf, 2025.

Bradley Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics: Methodology and distribution*, pages 569–593. Springer, 1992.

Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. Chapman and Hall/CRC, 1994.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Friedrich Götze and Alfredas Račkauskas. Adaptive choice of bootstrap sample sizes. *Lecture Notes-Monograph Series*, pages 286–309, 2001.

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

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*, 2023.

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

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. Making large language models better reasoners with step-aware verifier. *arXiv preprint arXiv:2206.02336*, 2022.

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

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

OpenAI. Introducing openai o1. https://openai.com/o1/, 2024. OpenAI Blog.

Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

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.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*, 2018.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL https://aclanthology.org/N19-1421.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

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

Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. Interpretable preferences via multi-objective reward modeling and mixture-of-experts. In *EMNLP*, 2024a.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

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. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024b.

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.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. An empirical analysis of compute-optimal inference for problem-solving with language models. 2024.

Rui Yang, Ruomeng Ding, Yong Lin, Huan Zhang, and Tong Zhang. Regularizing hidden states enables learning generalizable reward model for llms. *arXiv preprint arXiv:2406.10216*, 2024.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023. *URL https://arxiv.org/abs/2305.10601*, 3, 2023.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: Llm self-training via process reward guided tree search, 2024a. *URL https://arxiv. org/abs/2406.03816*.

List of Appendices

We provide a brief description of the material in the appendix of the paper.

- Appendix A provides theoretical results on the asymptotic behavior of BoN's output distribution and the proof for Theorem 4.1.
- Appendix B provides a closed-form calculation of bootstrapped BoN's output distribution for more efficient calculations.
- Appendix C investigates the effect of reward noise and base model on different algorithms in a synthetic setup.
- Appendix D provides extra details for the experiments and implementations.
- Appendix E provides additional experimental results.

A. Theoretical Results

In this section, we provide the formal theoretical results and the proof of Theorem 4.1. To do so, we first need to show the convergence of BoN's output distribution, which is done in Section A.1 and Theorem A.1. We prove Theorem 4.1 in Section A.2.

A.1. Asymptotic Behavior of BoN's Output Distribution

Theorem A.1. *For final answer z such that $\pi_{ref}(z) \in (0,1)$, let F_0 and F_1 represent continuous cumulative distribution functions (CDFs) of the conditional distributions $\mathbb{P}(r(Y)|f(Y)=z)$ and $\mathbb{P}(r(Y)|f(Y)\neq z)$, respectively. Define x_0 and x_1 to be right endpoint of them,*

$$x_0 \triangleq \sup\{x \in \mathbb{R} : F_0(x) < 1\}, \quad x_1 \triangleq \sup\{x \in \mathbb{R} : F_1(x) < 1\}.$$

As $N \to \infty$, if

(i) $x_0 < x_1$, *we have* $\pi_N(z) \to 0$.

(ii) $x_0 > x_1$, *we have* $\pi_N(z) \to 1$.

(iii) $x_0 = x_1 = x^*$, *assume for* $c \in [0,\infty]$, *we have*

$$\lim_{x \uparrow x^*} \frac{1-F_0(x)}{1-F_1(x)} = c, \tag{1}$$

then,

$$\pi_N(z) \to \frac{c \cdot \pi_{ref}(z)}{1+(c-1)\cdot \pi_{ref}(z)}.$$

Proof. We first define some random variables to better express $\pi_N(z)$. Assume we use F_0 and F_1 to generate i.i.d. samples $R_1^0, R_2^0, \ldots \overset{\text{i.i.d.}}{\sim} F_0$ and $R_1^1, R_2^1, \ldots \overset{\text{i.i.d.}}{\sim} F_1$. For $n \geq 1$, let S_n^0 and S_n^1 be the maximum of the first n samples from F_0 and F_1, that is,

$$S_n^0 \triangleq \max_{i=1,\ldots,n} R_i^0 \quad , \quad S_n^1 \triangleq \max_{i=1,\ldots,n} R_i^1.$$

Also, for outputs $Y_1, \ldots, Y_N$ let $Z_i = f(Y_i)$, N^0 be the number of outputs that reach the final answer z, and $N^1 = N - N^0$ be the number of outputs that do not reach the final answer z.

We can express $\pi_N(z)$ as

$$\begin{aligned}\pi_N(z) &= \sum_{z_{1:N}} \mathbb{P}\big(Z_N^{\text{Best}} = z | Z_{1:N} = z_{1:N}\big) \cdot \mathbb{P}(Z_{1:N} = z_{1:N}) \\ &= \sum_{z_{1:N}} \mathbb{P}\left(\max_{z_i = z} r(Y_i) > \max_{z_i \neq z} r(Y_i) | Z_{1:N} = z_{1:N}\right) \cdot \mathbb{P}(Z_{1:N} = z_{1:N}).\end{aligned} \tag{2}$$

Now, note that due $Y_1, \ldots, Y_N$ being i.i.d., we have

$$\mathbb{P}(r(Y_1), \ldots, r(Y_N)|Z_{1:N} = z_{1:N}) = \prod_i \mathbb{P}(r(Y_i)|Z_i = z_i).$$

By definition of R_i^0 and R_i^1, we can therefore write (2) as

$$\pi_N(z) = \sum_{z_{1:N}} \mathbb{P}\big(S_{N^0}^0 > S_{N^1}^1 | Z_{1:N} = z_{1:N}\big) \cdot \mathbb{P}(Z_{1:N} = z_{1:N}) = \mathbb{P}\big(S_{N^0}^0 > S_{N^1}^1\big).$$

For simplicity, we define $S^1 \triangleq S_{N^1}^1$ and $S^0 \triangleq S_{N^0}^0$. Now, we can express $\pi_N(z)$ as

$$\pi_N(z) = \mathbb{P}\big(S^0 > S^1\big).$$

Note that $S^0 \xrightarrow{d} x_0$ and $S^1 \xrightarrow{d} x_1$, which leads to the statement for cases *(i)* and *(ii)* straightforward. We focus on case *(iii)*. Let $\bar{F}_0(x) \triangleq 1 - F_0(x)$ and $\bar{F}_1(x) \triangleq 1 - F_1(x)$ be the complementary CDFs of F_0 and F_1, respectively. To quantify $\mathbb{P}\big(S^0 > S^1\big)$, we note that $\bar{F}_1$ is strictly decreasing in a neighborhood of S^1. Thus,

$$\lim_{N\to\infty} \pi_N(z) = \lim_{N\to\infty} \mathbb{P}\big(S^0 > S^1\big) = \lim_{N\to\infty} \mathbb{P}\big(N\bar{F}_1(S^0) < N\bar{F}_1(S^1)\big). \tag{3}$$

Therefore, we turn to study the joint distribution of $(N\bar{F}_1(S^0), N\bar{F}_1(S^1))$ as $N \to \infty$. This will be achieved by quantifying the distribution of $(n_0\bar{F}_1(S_{n_0}^0), n_1\bar{F}_1(S_{n_1}^1))$ as $n_0, n_1 \to \infty$ and relating it to the distribution of $(N\bar{F}_1(S^0), N\bar{F}_1(S^1))$.

Since F_1 is continuous, $F_1(R_i^1) \sim U[0,1]$ is uniformly distributed for any i. Define $U_i = \bar{F}_1(R_i^1) \sim U[0,1]$. It is well known that

$$n_1 \min_{i=1,\ldots,n_1} U_i \xrightarrow{d} \mathrm{Exp}(1) \qquad\qquad (n_1 \to \infty),$$

which due to $\min_i \bar{F}_1(R_i^1) = \bar{F}_1(S_{n_1}^1)$, translates to

$$n_1\bar{F}_1(S_{n_1}^1) \xrightarrow{d} \mathrm{Exp}(1) \qquad\qquad (n_1 \to \infty). \tag{4}$$

Similarly, we can show that $n_0\bar{F}_0(S_{n_0}^0) \xrightarrow{d} \mathrm{Exp}(1)$ as $n_0 \to \infty$. However, our goal is to analyze the distribution of $n_0\bar{F}_1(S_{n_0}^0)$. To do so, we use the tail-equivalence condition (1). We note that $S_{n_0}^0 \xrightarrow{d} x^*$, therefore, $\bar{F}_0(S_{n_0}^0)/\bar{F}_1(S_{n_0}^0) \xrightarrow{d} c$ as $n_0 \to \infty$. Together, we get

$$n_0\bar{F}_1(S_{n_0}^0) = \frac{n_0\bar{F}_0(S_{n_0}^0)}{\bar{F}_0(S_{n_0}^0)/\bar{F}_1(S_{n_0}^0)} \xrightarrow{d} \frac{\mathrm{Exp}(1)}{c} \qquad\qquad (n_0 \to \infty). \tag{5}$$

Due to the independence of $S_{n_1}^1$ and $S_{n_0}^0$, we can combine (4) and (5) to get

$$\big(n_0\bar{F}_1(S_{n_0}^0), n_1\bar{F}_1(S_{n_1}^1)\big) \xrightarrow{d} (E/c, F) \qquad\qquad (n_0, n_1 \to \infty),$$

where $E, F \overset{\text{i.i.d.}}{\sim} \mathrm{Exp}(1)$. As $N \to \infty$, we have $N^0, N^1 \xrightarrow{p} \infty$, therefore,

$$\big(N^0\bar{F}_1(S_{N^0}^0), N^1\bar{F}_1(S_{N^1}^1)\big) \xrightarrow{d} (E/c, F) \qquad\qquad (N \to \infty).$$

Finally, we use the fact that $N^0/N \xrightarrow{d} \pi_{\text{ref}}(z)$ and $N^1/N \xrightarrow{d} 1 - \pi_{\text{ref}}(z)$ to get

$$\big(N\bar{F}_1(S^0), N\bar{F}_1(S^1)\big) = \left(\frac{N^0\bar{F}_1(S_{N^0}^0)}{N^0/N}, \frac{N^1\bar{F}_1(S_{N^1}^1)}{N^1/N}\right) \xrightarrow{d} \left(\frac{E}{c\cdot\pi_{\text{ref}}(z)}, \frac{F}{1-\pi_{\text{ref}}(z)}\right). \tag{6}$$

Combined with (3), we conclude that

$$\lim_{N\to\infty} \pi_N(z) = \mathbb{P}\left(\frac{E}{c\cdot\pi_{\text{ref}}(z)} < \frac{F}{1-\pi_{\text{ref}}(z)}\right) = \frac{c\pi_{\text{ref}}(z)}{1-\pi_{\text{ref}}(z)+c\pi_{\text{ref}}(z)}.$$

□

A.2. Proof of Theorem 4.1

We restate Theorem 4.1 with the assumptions not included in the main text.

Theorem A.2. *Assume that there are finite possible values for Z and for every possible final answer z, the conditions of Theorem A.1 hold. If as $N \to \infty$, we have $m \to \infty$ and $m/N \to 0$, then for any $\epsilon > 0$, the estimated $\hat{\pi}_{m,N}$ will converge to the true distribution π_m. That is,*

$$\lim_{n\to\infty} \mathbb{P}\big(\|\hat{\pi}_{m,N} - \pi_m\|_1 \geq \epsilon\big) = 0.$$

Proof. Since there are finite possible values for Z, it suffices to show the convergence in estimated probability of each possible final answer z. We show that for any z, and $\epsilon > 0$, we have

$$\lim_{N\to\infty} \mathbb{P}(|\hat{\pi}_{m,N}(z) - \pi_m(z)| \geq \epsilon) = 0. \tag{7}$$

We use the result by Bickel et al. (2011, Equation 3.14) to show this claim. To do so, we first frame our problem in their notation. For $1 \leq i \leq N$, let $Z_i \triangleq f(Y_i)$ be (the one-hot encoding of) the final answer reached by Y_i, and $R_i \triangleq r(Y_i)$ be the numerical reward of Y_i. We define

$$X_i \triangleq (Z_i, R_i).$$

We define the bootstrap statistic of $X_1, \ldots, X_m$ as

$$T_m = \mathbb{I}\big[Z_m^{\text{Best}} = z\big] + \frac{D}{4} \sim L_m,$$

where $\mathbb{I}[\cdot]$ is the indicator function, $D \sim \text{Bernoulli}(0.5)$ is an independent Bernoulli random variable, and L_m is defined to be the distribution of T_m. Basically, T_m is the indicator of z being selected by BoN, plus a small random noise to ensure the non-degeneracy condition as $m \to \infty$. We define the function $h(t) = \mathbb{I}[t > 0.5]$, so that the parameter of interest θ_m becomes

$$\theta_m \triangleq \mathbb{E}h(T_m) = \pi_m(z),$$

as intended. Lastly, one can verify that since T_m is invariant of repetitions and permutations of its inputs $X_1, \ldots, X_m$, in our case, we have for any $0 < x < 1$,

$$\delta_m(x) \triangleq \big|\pi_{\lfloor mx \rfloor}(z) - \pi_m(z)\big|.$$

We now show the conditions of Bickel et al. (2011, Theorem 2). First, we need to show that L_m, the distribution of T_m, is convergent. According to Theorem A.1, we have

$$\lim_{m\to\infty} \pi_m(z) \triangleq \pi_\infty(z)$$

for some $\pi_\infty(z) \in [0, 1]$. Therefore, as $m \to \infty$, we have

$$L_m \xrightarrow{d} \text{Bernoulli}(\pi_\infty(z)) + \frac{\text{Bernoulli}(0.5)}{4}.$$

For condition Bickel et al. (2011, Equation 3.11) we need to show that for any $M < \infty$, we have

$$\delta_m(1 - xm^{-1/2}) \to 0$$

uniformly for all $0 < x < M$. By definition, it suffices to show that for any $0 < x < M$, we have

$$\big|\pi_{\lfloor m - x\sqrt{m} \rfloor}(z) - \pi_m(z)\big| \to 0.$$

This follows from the fact that $\pi_m(z)$ is convergent to $\pi_\infty(z)$. For any $\varepsilon > 0$, pick M_0 such that for any $m_0 \geq M_0$, we have

$$|\pi_{m_0}(z) - \pi_\infty(z)| < \frac{\varepsilon}{2},$$

and M_1 such that for any $M_1 - M\sqrt{M_1} \geq M_0$. Then for any $m \geq M_1$, we have

$$\left|\pi_{\lfloor m - x\sqrt{m} \rfloor}(z) - \pi_\infty(z)\right| < \varepsilon/2 \quad \text{and} \quad |\pi_m(z) - \pi_\infty(z)| < \varepsilon/2.$$

Together, we have

$$\left|\pi_{\lfloor m - x\sqrt{m} \rfloor}(z) - \pi_m(z)\right| < \varepsilon$$

and achieve the uniform convergence condition.

Finally, note that our statistic T_m is not dependent on the sampling distribution p_{ref} and Bickel et al. (2011, Equation 3.13) is satisfied. □

B. Closed-Form Calculation of Bootstrapped BoN's Output Distribution

In Section 4.2, we proposed approximating $\hat{\pi}_{m,N}$ by running BoN on a large number B of subsets of size m sampled with replacement from the N generated outputs. In practice, $B = 10,000$ is sufficient. This calculation is negligible compared to the generation of outputs from the LLM and can be carried out on a CPU. Nonetheless, we here show that it can also be done in $\mathcal{O}(N \log N)$.

Define $R_i = r(Y_i)$ for $1 \leq i \leq N$, and let $i_1, i_2, \ldots, i_N$ be such that

$$R_{i_1} < R_{i_2} < \ldots < R_{i_N}.$$

For simplicity, we assume no ties occur among the rewards. The key insight is that for any $1 \leq k \leq N$, the probability of Y_{i_k} being selected in a randomly sampled subset of m outputs can be calculated in closed-form. We note that Y_{i_k} is selected if the subset only includes outputs among $Y_{i_1}, \ldots, Y_{i_k}$, but is not limited to $Y_{i_1}, \ldots, Y_{i_{k-1}}$ (and therefore contains Y_{i_k}). We get

$$\mathbb{P}(Y_{i_k} \text{ is the output of BoN on a resampled subset}) = \left(\frac{k}{N}\right)^m - \left(\frac{k-1}{N}\right)^m$$

Thus, for any final answer z, the probability of it being selected in a subset is

$$\hat{\pi}_{m,N}(z) = \sum_{k:Z_{i_k}=z} \left(\frac{k}{N}\right)^m - \left(\frac{k-1}{N}\right)^m.$$

This procedure only requires sorting the outputs according to their rewards and therefore has complexity of $\mathcal{O}(N \log N)$.

C. Effect of Reward Noise and Base Model's Success Probability

In this section, we investigate the effect of the base model and reward noise on the success probability of SC, BoN, and MoB. We consider a synthetic setup for a TRUE/FALSE question, where the correct answer is TRUE. Let p be the success probability of the base model, which is the probability that the base model generates a solution reaching the correct final answer.

Assume r^{oracle} is an oracle reward model that always assigns the reward of 1 to solutions that reach the correct answer, and 0 otherwise:

$$r^{\text{oracle}}(Y) = \begin{cases} 1, & \text{if } f(Y) = \text{TRUE}, \\ 0, & \text{if } f(Y) = \text{FALSE}. \end{cases}$$

To investigate the effect of an imperfect reward model, we consider a noisy reward model r^{noisy} that is equal to the oracle reward plus an exponentially distributed noise:

$$r^{\text{noisy}}(Y) = r^{\text{oracle}}(Y) + \text{Exp}(1/\beta).$$

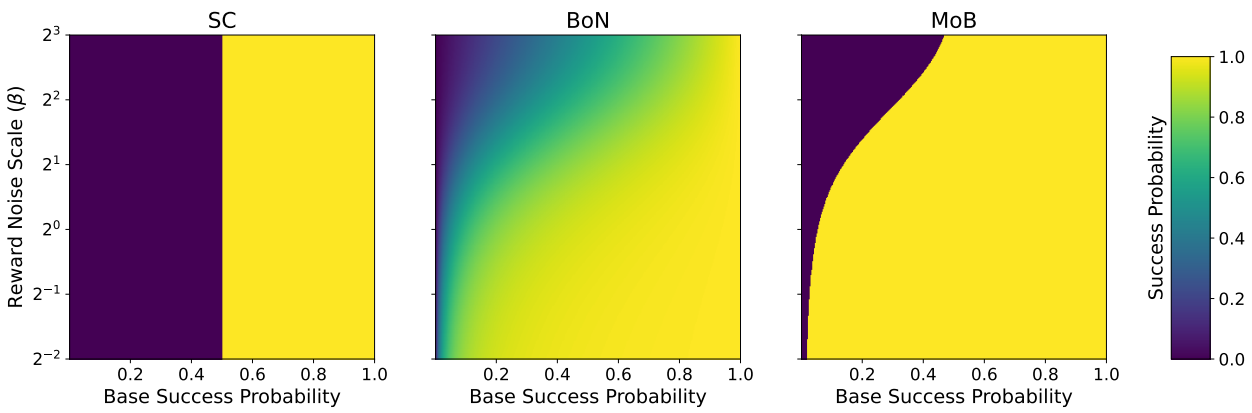

Figure 7: Success probability of SC, BoN, and MoB with infinite budget for different values of the base model's success probability and reward noise.

The parameter β controls the noise level, where a larger β indicates a noisier reward model. To see this, note that the expected value and the standard deviation of the noise are equal to β. If β is large, the noise will dominate the signal from the oracle reward, and the noisy reward model will be less informative.

We visualize the success probability of SC, BoN, and MoB with infinite budget $N = \infty$ in Figure 7. SC's success probability, as shown in the left plot of Figure 7, is independent of the reward noise. It is either equal to 1 when $p > 0.5$ (the correct answer is the most probable answer), or equal to 0 otherwise. For BoN, consider two extreme cases for the reward noise. When the reward model is perfect (β small), BoN's success probability is 1 regardless of the base model's success probability. This is shown in the bottom edge of the middle plot in Figure 7. In this case, BoN is preferable over SC. On the other hand, when the reward model is completely uninformative (β large), BoN's success probability is equal to the base model's success probability. This is shown in the top edge of the middle plot in Figure 7. As shown by Wang et al. (2022), SC has a higher accuracy over the base model and, in this case, BoN. MoB's success probability is equal to 1 if BoN's success probability is at least 0.5, as shown in the right plot of Figure 7. We see that MoB shows a similar behavior to SC when the reward model is uninformative, and when the reward model is perfect, MoB behaves like BoN.

In this setup, we can study the success probability of BoN and MoB with an infinite budget $N = \infty$ theoretically. BoN's success probability depends on the reward's noise level. It can be calculated from Theorem A.1 as

$$\text{BoN success probability with infinite budget} = \frac{e^{1/\beta}p}{1 - p + e^{1/\beta}p}.$$

Note that if the reward model is perfect ($\beta = 0$), both the numerator and denominator go to infinity, and we reach the success probability of 1. With $B = \infty$, the noise becomes dominant, and BoN's success probability remains equal to the base model p even with infinite budget. Due to Theorem 4.1, MoB solves the problem if the correct answer is BoN's most probable outcome. Therefore,

$$\text{MoB success probability with infinite budget} = \begin{cases} 1, & \text{if } \frac{e^{1/\beta}p}{1-p+e^{1/\beta}p} > 0.5, \\ 0, & \text{otherwise.} \end{cases}$$

This is favorable over BoN in scenarios where BoN still prefers the correct answer, as it can find the correct answer reliably without randomness.

D. Implementation and Experiment Details

In this section, we provide more details on how the experiments in the paper are conducted.

D.1. Evaluation Experiments

Benchmarks. We run our experiments on five popular benchmarks. MATH500, first introduced by Lightman et al. (2023), is a randomly sampled subset of 500 math questions with short final answers from the MATH dataset (Hendrycks et al., 2021). We use the math and chemistry questions from the MMLU-Pro benchmark (Wang et al., 2024b), which includes multiple-choice questions on a variety of topics. We also run our experiments on GSM8K (Cobbe et al., 2021a) that contains grade school math questions in short final answer format. Lastly, we use the CommonsenseQA benchmark (Talmor et al., 2019) that tests the model's commonsense reasoning through multiple-choice questions. For all benchmarks, we randomly select 500 questions for our experiments.

Implementation Details. In the implementation of MoB, we always use the closed-form calculation of $\hat{\pi}_{m,N}$ discussed in Appendix B to efficiently perform the bootstrap estimate. Therefore, in the actual implementation, there is no parameter B and we effectively operate as if $B = \infty$ was chosen. We use Huggingface's Python library for all the output generations. We always use temperature 1 for inference and no extra modification of the next-token sampling procedure. The final answer extraction, evaluation, and standard errors are calculated using the Language Model Evaluation Harness (Gao et al., 2024). For each question, we generate 512 outputs, and for each budget size N, we run each algorithm $\lfloor 512/N \rfloor$ times to provide better standard errors for the accuracies. For GSM8K, we use a 5-shot prompt. For MATH and MMLU-Pro questions, we use the zero-shot chain-of-thought prompting used in the official Llama3.1 models evaluation (Grattafiori et al., 2024) on MATH (Hendrycks et al., 2021). This prompt and the prompt used for CommonsenseQA are given in the following.

Prompt for MATH and MMLU-Pro

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

Prompt for CommonsenseQA

```
Use commonsense to solve the following multiple choice question. First
explain your solution and then give the final answer. Always finish your
answer with "the answer is (X)" where X is the correct letter choice.
Question:: <problem from dataset>
```

D.2. Experiments for Motivation, Oracle MoB, and Selection of Bootstrap Subset Size

In Figure 2, we discussed the success probability of BoN, which requires an estimate of BoN's output distribution. We use the same technique as in MoB to estimate this output distribution. To minimize the error of this approximation, we specifically generate 1,400 outputs for the math problems in MMLU-Pro with `Qwen2.5-3b-instruct` . Then, we use $\hat{\pi}_{N,1400}$, as defined in Section 4.2 as an estimate for π_N. Same technique is used in Figure 3 where the mode of $\hat{\pi}_{N,1400}$ is chosen as the output of oracle MoB, and Figure 4 to where the distribution estimation error is calculated with respect to $\hat{\pi}_{m,1400}$ instead of the true π_m.

In Figure 5, we consider seven fixed schedules for m, specifically $m = \lfloor N^\alpha \rfloor$ for $\alpha = 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8$. At any budget N, we compared the accuracy of MoB with adaptive m against the highest accuracy among the seven instantiations of fixed schedule MoB.

E. Additional Experimental Results

In this section, we provide additional experimental results for all 30 setups.

E.1. Adaptive Subset Size Selection

In Section 4, we compared MoB with adaptive choice of m with the optimal choice of m. We provide this comparison in MATH500 (Figure 8), MMLU-Pro-Math (Figure 9), MMLU-Pro-Cham (Figure 10), GSM8K (Figure 11), and CommonsenseQA (Figure 12).

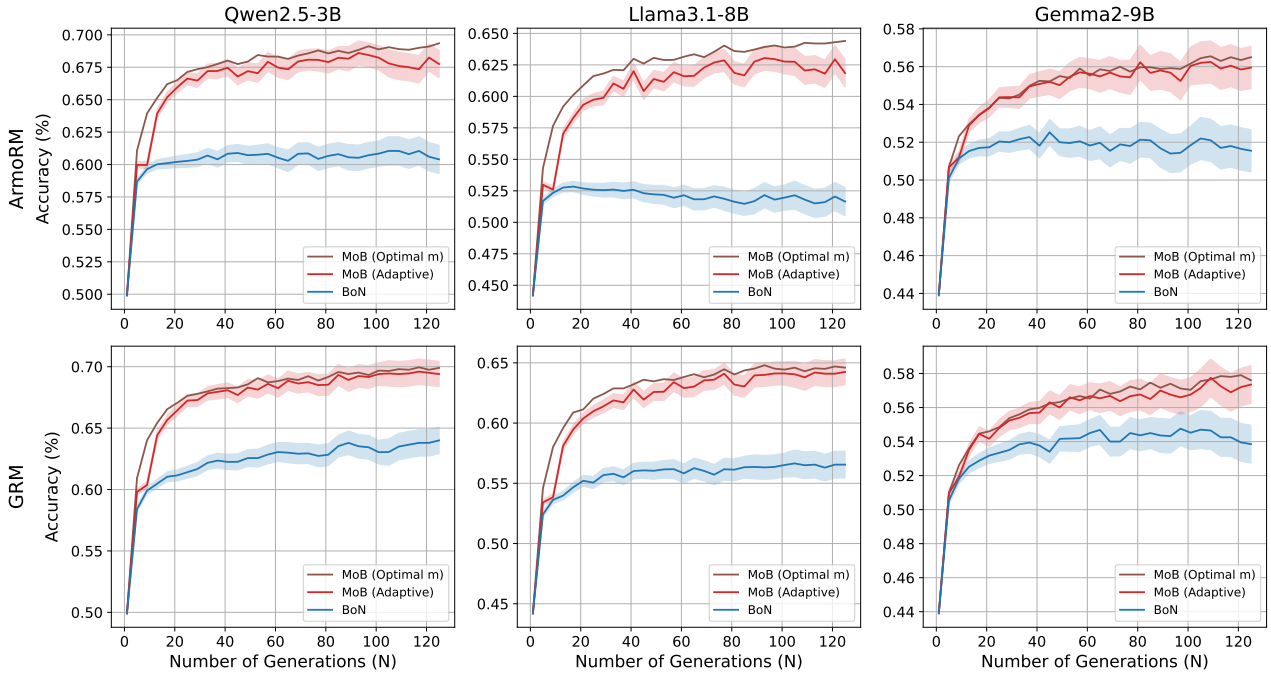

Figure 8: Comparison of MoB with adaptive m against MoB with optimal m on the MATH500 dataset with ArmoRM *(Up)* and GRM *(Down)* reward models, and Qwen2.5-3B *(Left)*, Llama3.1-8B *(Middle)*, and Gemma2-9B *(Right)* base models. Shaded areas show standard error.

E.2. Evaluation Experiments

We compare MoB with baselines in MATH500 (Figure 13, Table 3), MMLU-Pro-Math (Figure 14, Table 4), MMLU-Pro-Cham (Figure 15, Table 5), GSM8K (Figure 16, Table 6), and CommonsenseQA (Figure 17, Table 7).

Table 3: Results on MATH500 across all base and reward models ($N = 128$).

	ArmoRM			**GRM**		
	Gemma-2-9B	LLaMA3.1-8B	Qwen2.5-3B	Gemma-2-9B	LLaMA3.1-8B	Qwen2.5-3B
BoN	52.20±1.12	51.65±1.12	60.50±1.09	53.85±1.11	56.55±1.11	64.15±1.07
SC	52.65±1.12	61.15±1.09	66.30±1.06	52.65±1.12	61.15±1.09	66.30±1.06
WBoN	53.60±1.12	**63.10±1.08**	67.05±1.05	56.00±1.11	**63.65±1.08**	67.30±1.05
MoB (Ours)	**56.65±1.11**	**62.50±1.08**	**68.25±1.04**	**57.80±1.10**	**64.15±1.07**	**69.80±1.03**
↑MoB over BoN	4.45±1.57	10.85±1.56	7.75±1.51	3.95±1.57	7.60±1.54	5.65±1.48

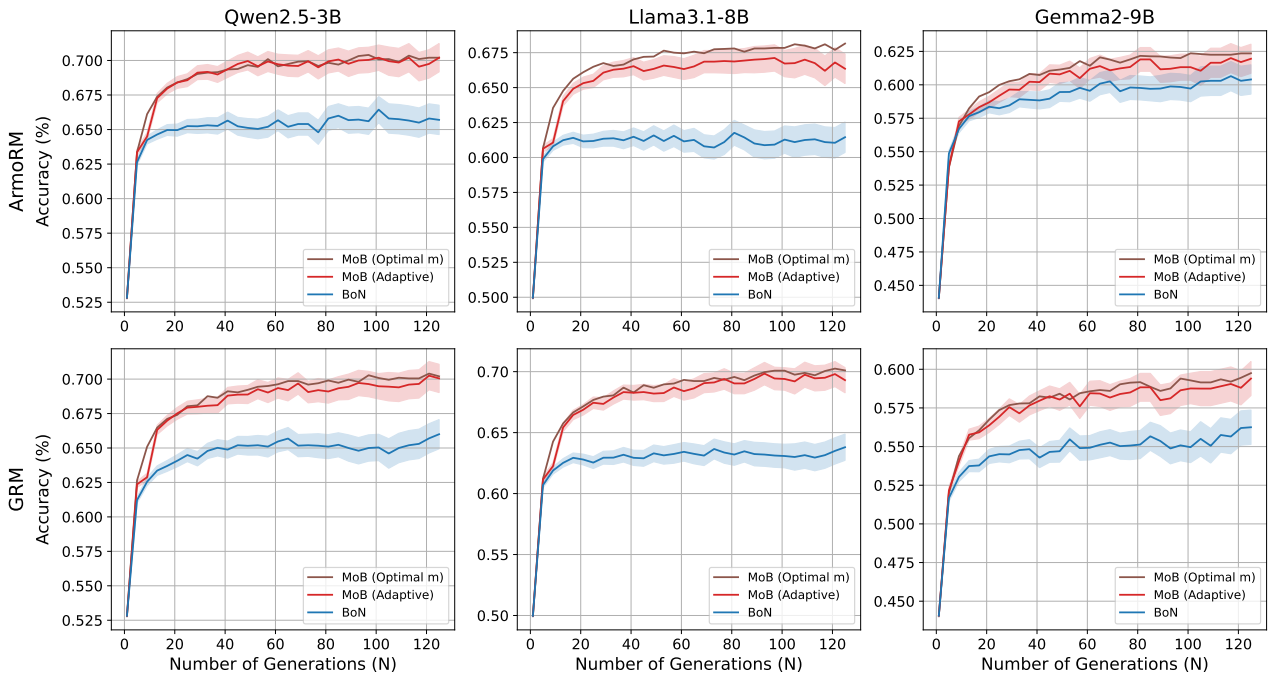

Figure 9: Comparison of MoB with adaptive m against MoB with optimal m on the MMLU-Pro-Math dataset with ArmoRM *(Up)* and GRM *(Down)* reward models, and Qwen2.5-3B *(Left)*, Llama3.1-8B *(Middle)*, and Gemma2-9B *(Right)* base models. Shaded areas show standard error.

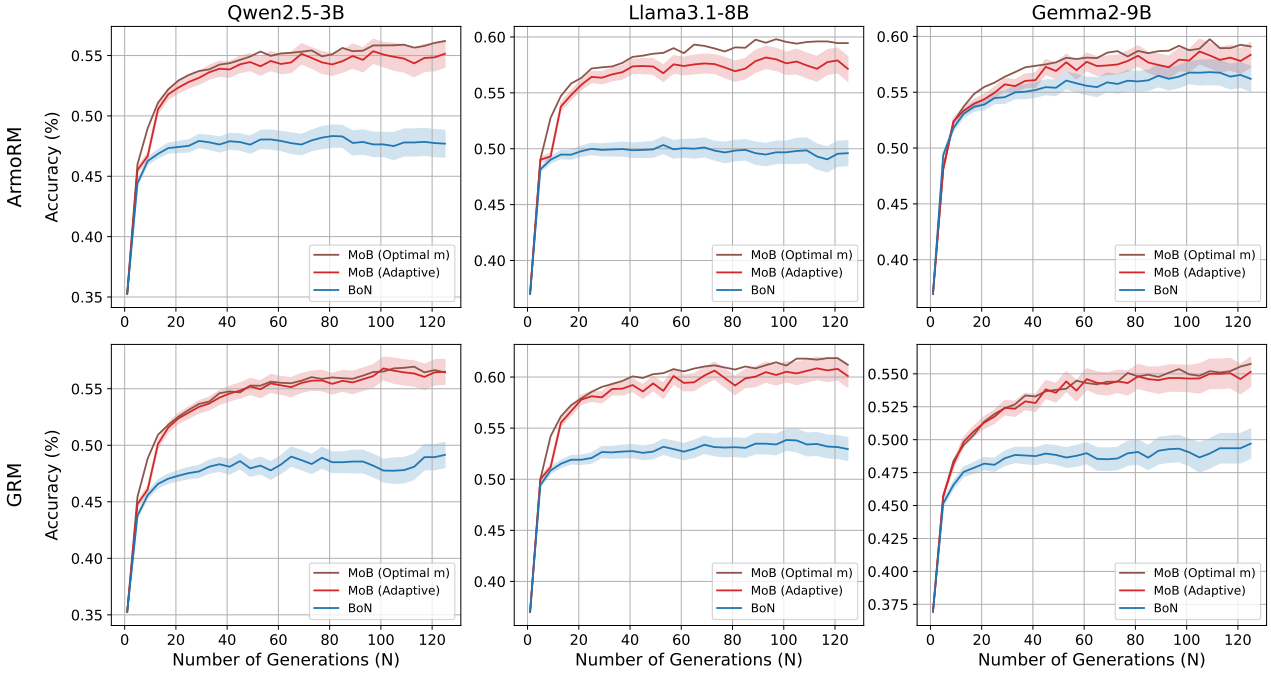

Figure 10: Comparison of MoB with adaptive m against MoB with optimal m on the MMLU-Pro-Chem dataset with ArmoRM *(Up)* and GRM *(Down)* reward models, and Qwen2.5-3B *(Left)*, Llama3.1-8B *(Middle)*, and Gemma2-9B *(Right)* base models. Shaded areas show standard error.

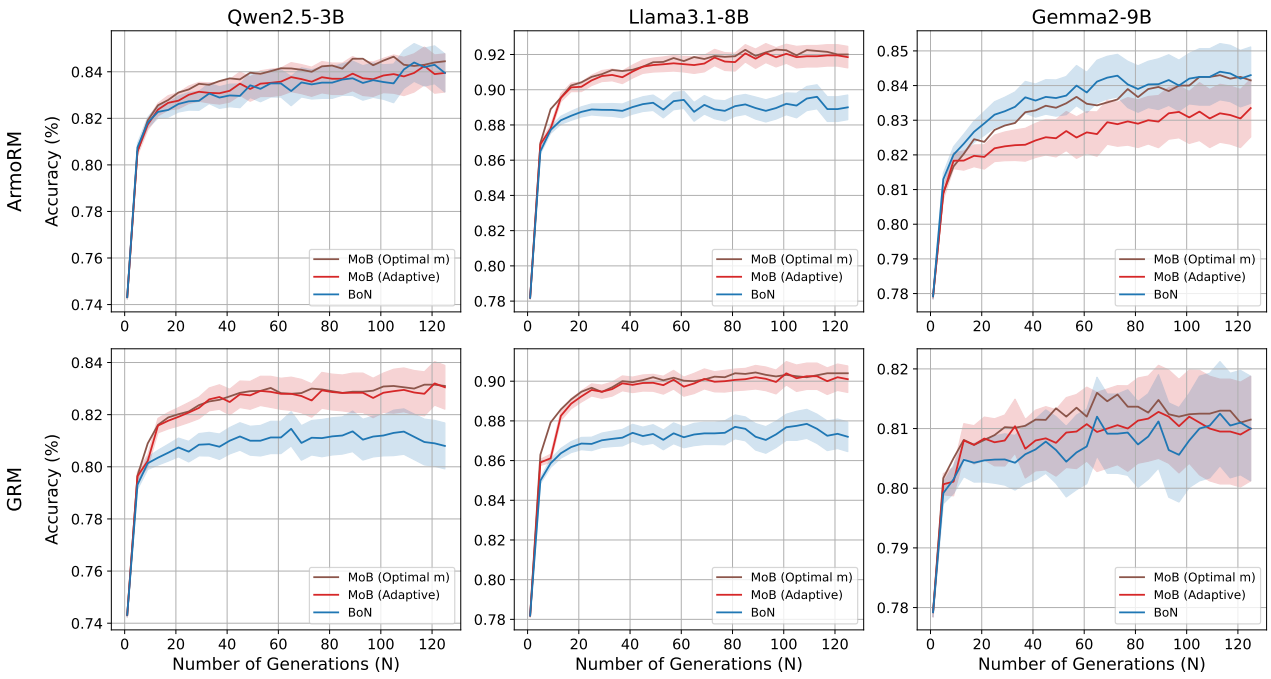

Figure 11: Comparison of MoB with adaptive m against MoB with optimal m on the GSM8K dataset with ArmoRM *(Up)* and GRM *(Down)* reward models, and Qwen2.5-3B *(Left)*, Llama3.1-8B *(Middle)*, and Gemma2-9B *(Right)* base models. Shaded areas show standard error.

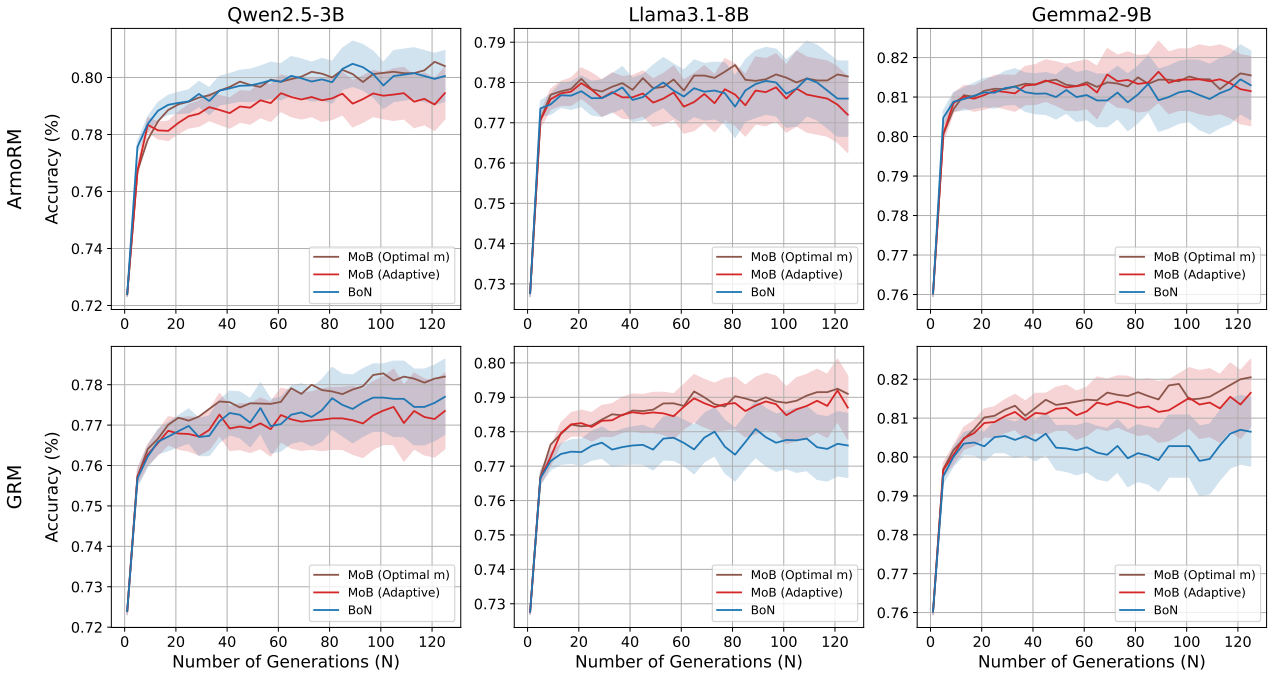

Figure 12: Comparison of MoB with adaptive m against MoB with optimal m on the CommonsenseQA dataset with ArmoRM *(Up)* and GRM *(Down)* reward models, and Qwen2.5-3B *(Middle)*, and Gemma2-9B *(Right)* base models. Shaded areas show standard error.

Table 4: Results on MMLU-Pro-Math across all base and reward models ($N = 128$).

	ArmoRM			GRM		
	Gemma-2-9B	LLaMA3.1-8B	Qwen2.5-3B	Gemma-2-9B	LLaMA3.1-8B	Qwen2.5-3B
BoN	60.45±1.09	61.40±1.09	65.95±1.06	56.15±1.11	64.10±1.07	66.10±1.06
SC	49.95±1.12	62.95±1.08	65.60±1.06	49.95±1.12	62.95±1.08	65.60±1.06
WBoN	52.25±1.12	**66.45±1.06**	66.70±1.05	56.45±1.11	60.05±1.10	64.35±1.07
MoB (Ours)	**61.55±1.09**	**66.70±1.05**	**69.80±1.03**	**59.35±1.10**	**69.05±1.03**	**69.30±1.03**
↑MoB over BoN	1.10±1.54	5.30±1.52	3.85±1.48	3.20±1.56	4.95±1.49	3.20±1.48

Table 5: Results on MMLU-Pro-Chem across all base and reward models ($N = 128$).

	ArmoRM			GRM		
	Gemma-2-9B	LLaMA3.1-8B	Qwen2.5-3B	Gemma-2-9B	LLaMA3.1-8B	Qwen2.5-3B
BoN	56.60±1.11	49.70±1.12	48.05±1.12	49.25±1.12	53.05±1.12	49.00±1.12
SC	43.45±1.11	50.35±1.12	52.50±1.12	43.45±1.11	50.35±1.12	52.50±1.12
WBoN	45.45±1.11	**57.65±1.11**	53.30±1.12	**57.25±1.11**	49.75±1.12	53.10±1.12
MoB (Ours)	**58.05±1.10**	**57.40±1.11**	**54.75±1.11**	54.60±1.11	**60.75±1.09**	**56.45±1.11**
↑MoB over BoN	1.45±1.56	7.70±1.57	6.70±1.58	5.35±1.58	7.70±1.56	7.45±1.57

Table 6: Results on GSM8K across all base and reward models ($N = 128$).

	ArmoRM			GRM		
	Gemma-2-9B	LLaMA3.1-8B	Qwen2.5-3B	Gemma-2-9B	LLaMA3.1-8B	Qwen2.5-3B
BoN	**84.20±0.82**	89.00±0.70	**83.85±0.82**	**81.20±0.87**	87.15±0.75	80.95±0.88
SC	80.55±0.89	88.10±0.72	80.40±0.89	**80.55±0.89**	88.10±0.72	80.40±0.89
WBoN	80.75±0.88	88.70±0.71	81.10±0.88	79.45±0.90	77.75±0.93	81.25±0.87
MoB (Ours)	83.30±0.83	**91.75±0.62**	**83.85±0.82**	**81.15±0.87**	**90.50±0.66**	**82.85±0.84**
↑MoB over BoN	-0.90±1.17	2.75±0.93	0.00±1.16	-0.05±1.24	3.35±1.00	1.90±1.22

Table 7: Results on CommonsenseQA across all base and reward models ($N = 128$).

	ArmoRM			GRM		
	Gemma-2-9B	LLaMA3.1-8B	Qwen2.5-3B	Gemma-2-9B	LLaMA3.1-8B	Qwen2.5-3B
BoN	**81.20±0.87**	**77.80±0.93**	**80.15±0.89**	80.55±0.89	**78.05±0.93**	**77.70±0.93**
SC	79.25±0.91	75.75±0.96	76.20±0.95	79.25±0.91	75.75±0.96	76.20±0.95
WBoN	80.05±0.89	76.75±0.94	76.60±0.95	49.80±1.12	36.35±1.08	54.90±1.11
MoB (Ours)	**81.20±0.87**	**77.40±0.94**	**79.40±0.90**	**81.45±0.87**	**78.45±0.92**	**77.40±0.94**
↑MoB over BoN	0.00±1.24	-0.40±1.32	-0.75±1.27	0.90±1.24	0.40±1.30	-0.30±1.32

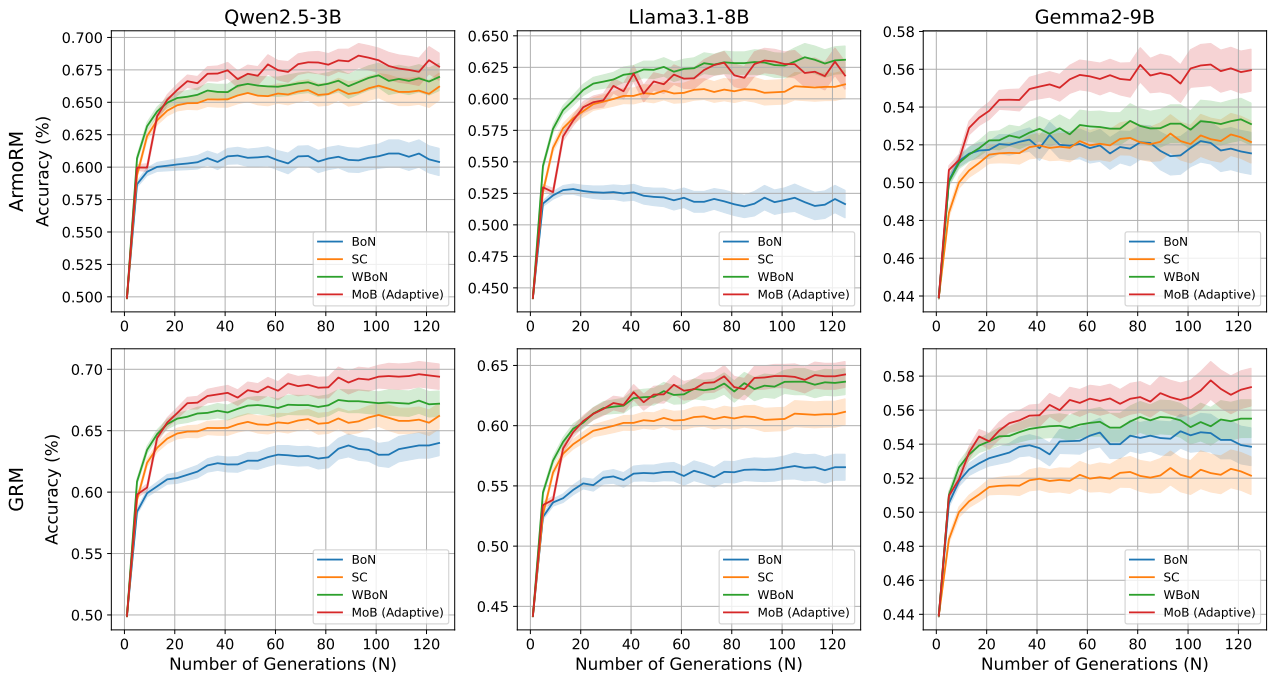

Figure 13: Comparison of MoB with the baselines on the MATH500 dataset with ArmoRM *(Up)* and GRM *(Down)* reward models, and Qwen2.5-3B *(Left)*, Llama3.1-8B *(Middle)*, and Gemma2-9B *(Right)* base models. Shaded areas show standard error.

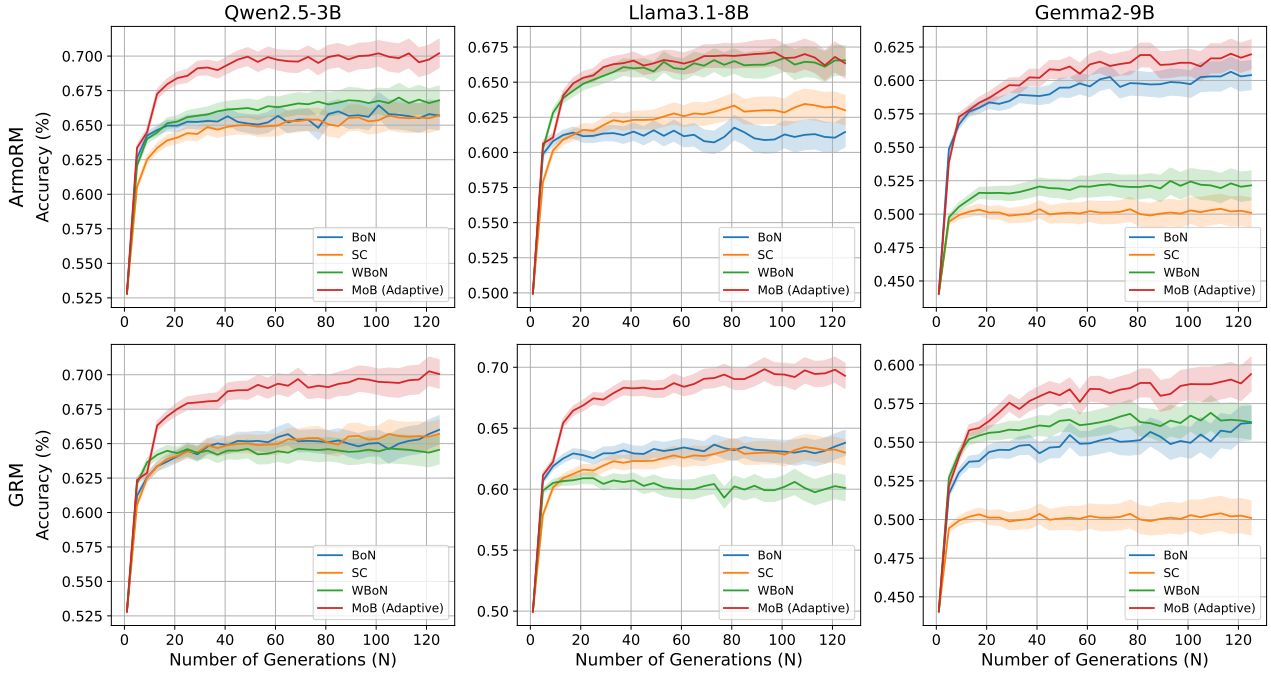

Figure 14: Comparison of MoB with the baselines on the MMLU-Pro-Math dataset with ArmoRM *(Up)* and GRM *(Down)* reward models, and Qwen2.5-3B *(Left)*, Llama3.1-8B *(Middle)*, and Gemma2-9B *(Right)* base models. Shaded areas show standard error.

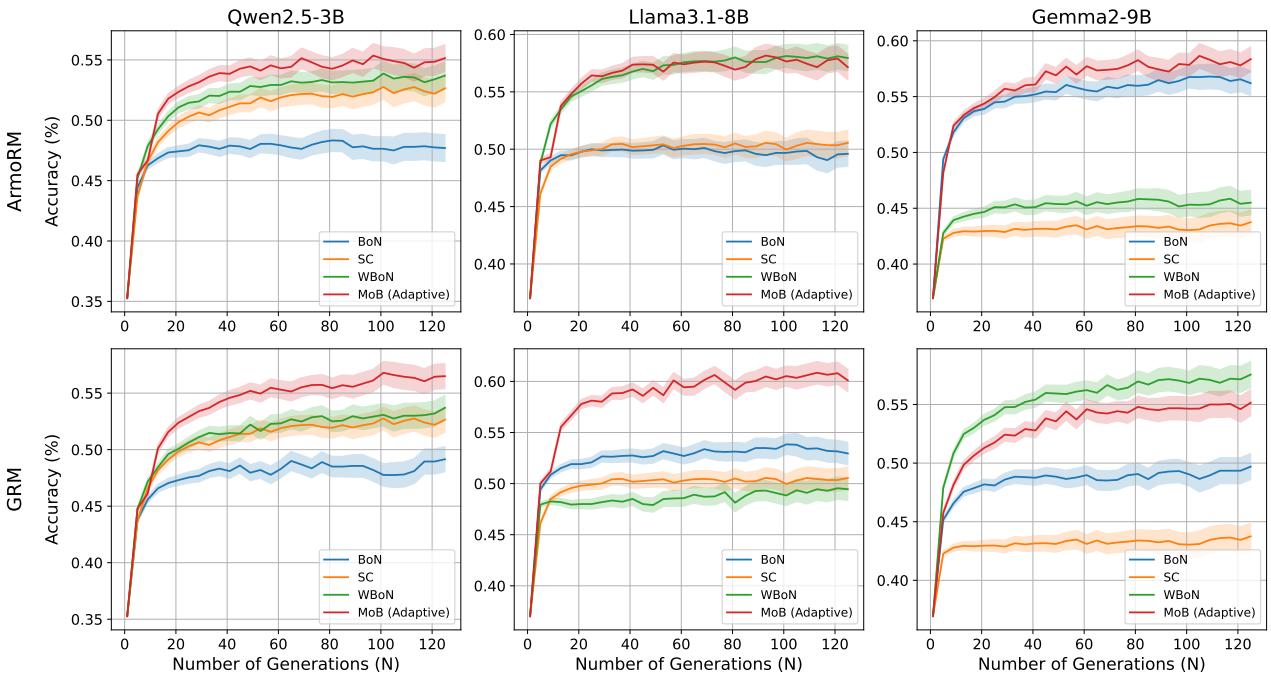

Figure 15: Comparison of MoB with the baselines on the MMLU-Pro-Chem dataset with ArmoRM *(Up)* and GRM *(Down)* reward models, and Qwen2.5-3B *(Left)*, Llama3.1-8B *(Middle)*, and Gemma2-9B *(Right)* base models. Shaded areas show standard error.

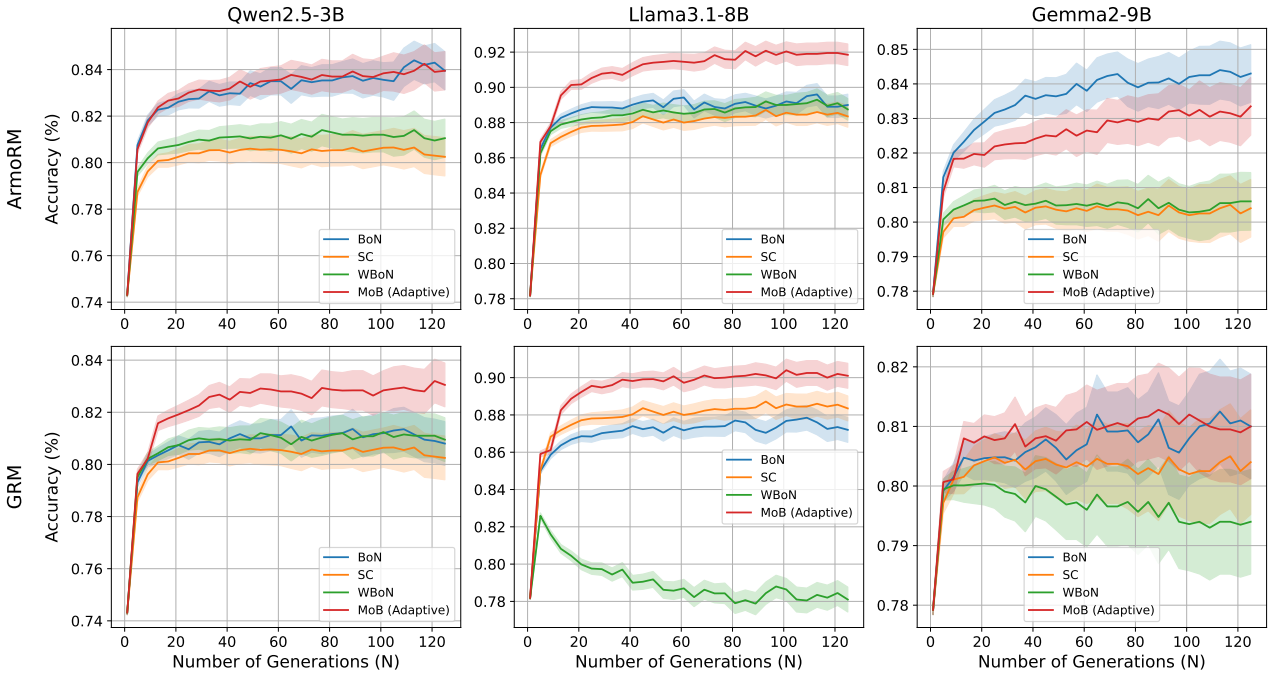

Figure 16: Comparison of MoB with the baselines on the GSM8K dataset with ArmoRM *(Up)* and GRM *(Down)* reward models, and Qwen2.5-3B *(Left)*, Llama3.1-8B *(Middle)*, and Gemma2-9B *(Right)* base models. Shaded areas show standard error.

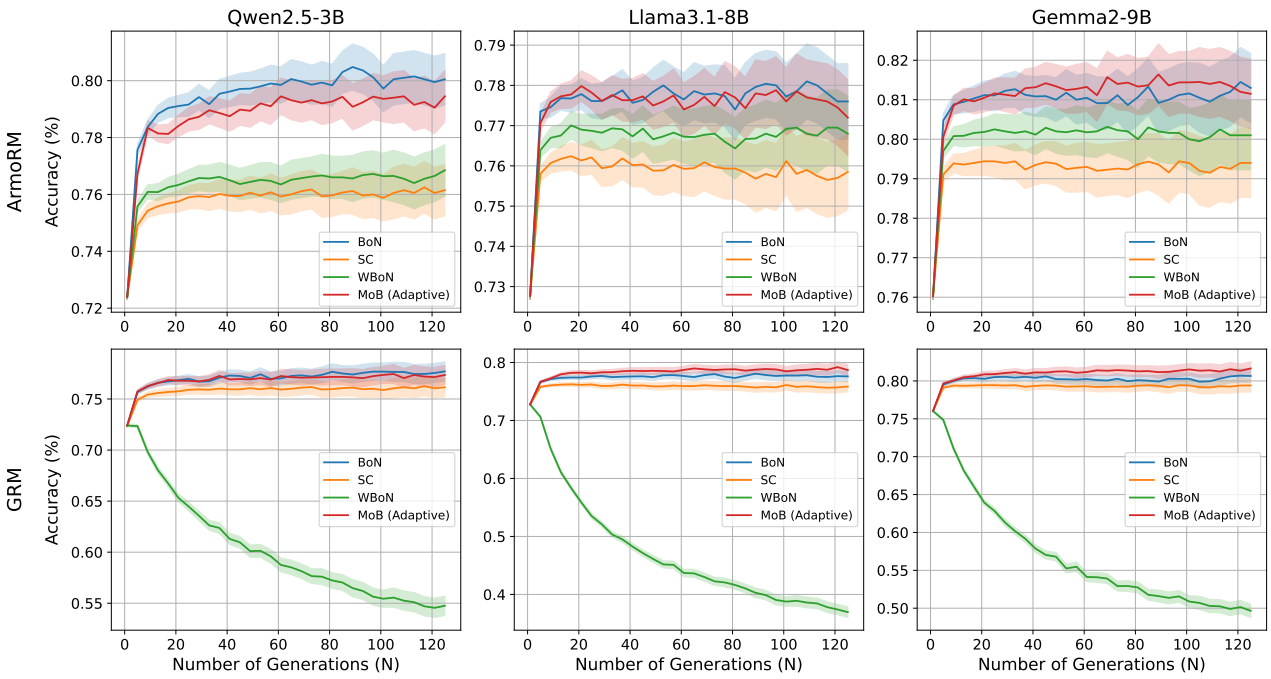

Figure 17: Comparison of MoB with the baselines on the CommonsenseQA dataset with ArmoRM *(Up)* and GRM *(Down)* reward models, and Qwen2.5-3B *(Left)*, Llama3.1-8B *(Middle)*, and Gemma2-9B *(Right)* base models. Shaded areas show standard error.