SANE: SELECTOR-AGNOSTIC NONPARAMETRIC EARLY STOPPING FOR EFFICIENT REPEATED SAMPLING

Anonymous authors
Paper under double-blind review

ABSTRACT

Repeated sampling improves the accuracy of large language models by generating multiple outputs from the model and selecting one of them as the final answer. Using a fixed sampling budget wastes computation on the easier tasks that require fewer samples. Early-stopping algorithms address this issue by stopping the sampling process once they detect additional samples are unlikely to improve the final answer. Existing algorithms are limited to the single selection mechanism they are designed for, such as self-consistency (SC) or best-of-N (BoN), and some require modeling the underlying distribution of rewards or answers, which may be challenging. We introduce SANE, a practical early-stopping algorithm for repeated sampling that is nonparametric and compatible with any black-box selector that satisfies a union-consistency property: if the selector's choice is the same for two disjoint sets of samples, its choice remains the same over their union. This is satisfied by the commonly used SC, BoN, and weighted BoN algorithms. After sampling n outputs, we lower bound the probability that the selector's choice remains the same with M additional samples. SANE bounds this probability by partitioning the $n + M$ samples into disjoint blocks and estimating the selector's choice for each of them through subsampling. We empirically show that SANE provides a superior cost-accuracy trade-off across a variety of tasks, models, reward models, and selector algorithms and can match the accuracy of fixed-budget sampling with 42%–73% fewer samples.

1 INTRODUCTION

Scaling test-time computation is an effective paradigm to improve the capabilities of Large Language Models (LLM) on a wide variety of tasks spanning mathematical reasoning, question-answering, and coding (Snell et al., 2025; Brown et al., 2024; OpenAI, 2024). A simple and popular approach for such scaling is repeated sampling, in which we sample multiple outputs from the model for a given task and aggregate them into a higher-quality final answer. Best-of-N (Lightman et al., 2023, BoN), self-consistency (Wang et al., 2022, SC), and weighted BoN (Li et al., 2022, WBoN) are early methods that popularized this technique. Beyond these, a growing number of more advanced repeated sampling methods have been introduced that improve the effectiveness of this approach. Recent work on avoiding reward hacking (Jinnai et al., 2024; Ichihara et al., 2025; Huang et al., 2025a), using the rankings according to rewards (Rakhsha et al., 2025), and even training models to aggregate the sampled outputs (Zhao et al., 2025) are some of the new methods developed.

One of the most critical drawbacks of repeated sampling is its computational cost. Sampling outputs from LLMs, especially the reasoning models, is notoriously computationally expensive (Samsi et al., 2023) and generating multiple outputs for repeated sampling dramatically worsens the computational burden. One solution to tackle this challenge is to adapt the number of generated outputs to the task in hand. In the standard practice of having a fixed sampling budget, some computation is wasted on generating outputs that do not improve the algorithm's final answer. When the outputs are aggregated through SC or BoN, early-stopping algorithms have been developed to detect these cases and stop the sampling process (Aggarwal et al., 2023; Li et al., 2024; Komiyama et al., 2026; Cordero-Encinar & Duncan, 2025; Huang et al., 2026; Wan et al., 2025b; Kalayci et al., 2025).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

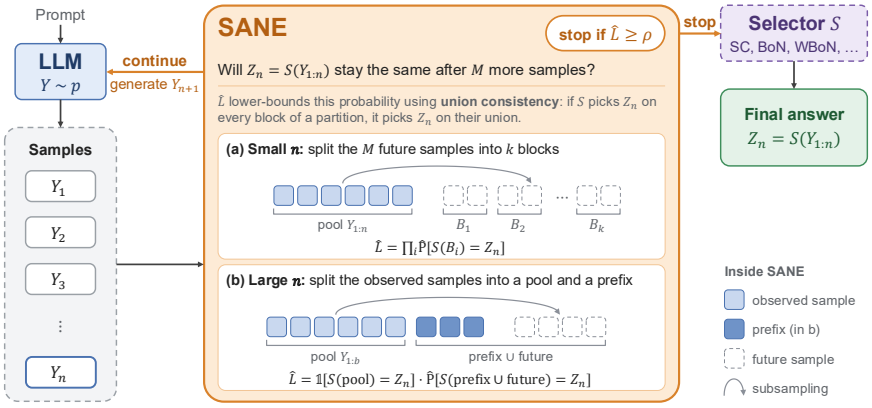


Figure 1: Overview of algorithm. After observing n samples, SANE derives a lower bound for the probability that the answer remains the same with M more samples. Sampling is stopped if the lower bound is larger than a threshold.

Most early-stopping algorithms rely on parametrically modeling the distribution of final answers generated by the LLM or the rewards obtained from the reward model to decide how sampling more outputs will change the SC or BoN's returned answer. This is an effective approach for the categorical distribution of LLM's final answers but becomes difficult for rewards or other quantities such as LLM's log probabilities that more advanced repeated sampling algorithms use. Furthermore, explicitly modeling how future samples will change the selected answer is challenging for some selection mechanisms. Therefore, early stopping algorithms have been limited to SC and BoN and are not available for the growing number of repeated sampling methods. In this paper, we address this issue by introducing **S**elector-**A**gnostic **N**onparametric **E**arly-stopping (SANE), an early stopping algorithm that provides adaptive-budgeting on a large family of black-box selection mechanisms. We show that a mild union-consistency condition for the selector, which is satisfied by SC, BoN, and WBoN, is enough to detect early-stopping.

Our union-consistency condition requires that if the selector's chosen final answer is the same for two disjoint sets of sampled outputs, it remains the same for their union. Assume n outputs $Y_1, \ldots, Y_n$ have been sampled from the LLM and our selector S has picked the final answer $Z_n = S(Y_{1:n})$ from them. We derive a lower bound on the *stability probability* that given $Y_{1:n}$, if the next M outputs $Y_{n+1}, \ldots, Y_{n+M}$ are observed, the selector's new choice $S(Y_{1:n+M})$ will remain the same as Z_n. The algorithm decides to stop sampling if this lower bound, and consequently the stability probability, is over a threshold. This avoids unnecessary sampling in cases that further sampling is unlikely to change the final outcome. The core idea to obtain this lower bound is to first partition $Y_{1:n+M}$ into disjoint subsets $B_1, \ldots, B_k$. Then, we note that due to the union-consistency assumption, if the selector's choice $S(B_i)$ equals Z_n for every B_i, then the choice over all $n + M$ samples is also Z_n. Therefore, the conditional probability $\mathbb{P}\big(\forall i : S(B_i) = Z_n \mid Y_{1:n}\big)$ of Z_n being picked from all blocks serves as a lower bound for the stability probability. We propose a partitioning scheme such that each conditional probability $\mathbb{P}\big(S(B_i) = Z_n \mid Y_{1:n}\big)$ can be estimated by subsampling from the observed outputs. Figure 1 provides an overview of the algorithm.

We provide a version of SANE with the theoretical guarantees that shows if the algorithm decides to stop sampling, the conditional expected accuracy gain of further sampling is bounded. Empirically, we evaluate SANEon five benchmarks, three base LLMs, three reward models, and three selection algorithms. We observe that SANEis even more effective than some specialized early stopping algorithms and requires 42% – 73% fewer samples to match the accuracy of the fixed sample budget of 16 samples.

2 Related Work

Prior work in early stopping has largely focused on self-consistency. Li et al. (2024) introduces Early-stopping Self-consistency that heuristically stops sampling once a fixed-size window of iden-

tical answers is observed. Adaptive-Consistency (Aggarwal et al., 2023) uses Bayesian inference with Beta or Dirichlet priors to model the LLM's answer distribution and stop once the distribution's mode becomes clear. Huang et al. (2026) extends this approach to using informative priors. Other statistical approaches use martingale certificates (Cordero-Encinar & Duncan, 2025) and Bayes factor (Komiyama et al., 2026). Reasoning-Aware Self-Consistency (Wan et al., 2025a) trains a model to evaluate the outputs and stops once enough high-scoring outputs have been generated, and CaTS (Huang et al., 2025b) distills confidence scores that are used both for stopping and aggregation. Difficulty-Adaptive Self-Consistency (Wang et al., 2025) ranks the questions in the dataset to pre-allocate the budget for each question accordingly.

Beyond self-consistency, early-stopping for Best-of-N has also been studied. Kalayci et al. (2025) fits a shifted exponential distribution to the reward values above the median and uses the Pandora box formulation for stopping. BEACON (Wan et al., 2025b) models the rewards by normal distribution through Bayesian inference to detect confidence in the selected answer. Damani et al. (2024) train an auxiliary model to predict marginal reward gains, and AdaBoN (Raman et al., 2026) models the reward distributions using Kernel Density Estimation, both to allocate the sampling budget across the dataset. These prior methods focus on a single selection algorithm, SC or BoN, and use statistical or learned signals tailored to that algorithm for stopping. In the contrary, SANEis a common early-stopping algorithm for any union-consistent selection including BoN, SC, and WBoN. It uses subsampling to estimate a lower bound for the stability probability, requires only black-box access to the selector, and does not need any separate training.

3 Background

In this section, we present the formal notations of a task, the formulation for repeated sampling and the selectors we will consider, as well as the early stopping problem for offline dataset processing.

Task definition. Each task is described by a prompt $x \in \mathcal{X}$. For the most part, we will fix the prompt x and suppress all dependencies on it. Given the prompt, the LLM is used to sample an output $Y \sim p$. This output $Y \in \mathcal{Y}$ is typically the whole sequence of tokens generated autoregressively by the LLM and for example, contains the reasoning trace and the chain-of-thought. The stochasticity of the output is due to the positive sampling temperature for sampling. We consider tasks with a finite space of final answers $\mathcal{Z}$. This could be the options in a multiple-choice question or the set of short mathematical expressions that can be the final answer for a math question. Any output $y \in \mathcal{Y}$ points to a final answer $z = f(y)$. We are interested in finding the correct answer z^*. The accuracy of a process that returns the (random) final answer Z for the prompt is $\mathbb{P}(Z = z^*)$.

Repeated sampling. In the repeated sampling approach, given a sample budget of N, we independently sample N outputs $Y_1, Y_2, \ldots, Y_N$ from the LLM's output distribution p. Then a permutation-invariant selector $S\colon \mathcal{Y}^* \to \mathcal{Z}$ decides the returned final answer $S(Y_{1:N}) \in \mathcal{Z}$. The selection operation can be arbitrarily complex and include using a reward model or the LLM itself. Given outputs $y_{1:k}$, BoN evaluates each output using a reward model $r\colon \mathcal{Y} \to \mathbb{R}$ and picks the one with the highest reward. Self-consistency, also called majority voting, picks the final answer that appears most frequently. Specifically,

$$S_{\mathrm{BoN}}(y_{1:k}) = f\big(\operatorname*{argmax}_{y \in y_{1:k}} r(y)\big) \qquad , \qquad S_{\mathrm{SC}}(y_{1:k}) = \operatorname*{argmax}_{z \in \mathcal{Z}} \sum_{i=1}^{k} \mathbf{1}\big[f(y_i) = z\big],$$

where $\mathbf{1}[\,\cdot\,]$ is the indicator function. Lastly, WBoN, sometimes called weighted majority-voting, reweights the votes by the rewards given by the reward model. That is,

$$S_{\mathrm{WBoN}}(y_{1:k}) = \operatorname*{argmax}_{z \in \mathcal{Z}} \sum_{i=1}^{k} \mathbf{1}\big[f(y_i) = z\big] r(y_i).$$

The weighting scheme of WBoN is only reasonable when the rewards are non-negative. In our experiments, we will ensure this by applying the sigmoid function to the values returned from the reward model.

Offline dataset processing and early stopping. We consider repeated sampling for the setting in which a dataset $\mathcal{D} \subseteq \mathcal{X}$ of prompts is given. Our goal is to find a (random) final answer Z_x for each prompt $x \in \mathcal{D}$. In this setting, the set of final answers $\{Z_x\}_{x\in\mathcal{D}}$ is returned all at once. The accuracy over the dataset is the expected number of tasks completed successfully, and the cost is measured by the expected average number of outputs sampled from the LLM per task. Sampling the outputs for a batch of prompts can be done in parallel, where a single output is generated for each prompt in the batch. This allows us to exploit the computational efficiency of batched LLM inference for the maximum sampling throughput while the samples for each individual task are generated sequentially.

Since the outputs for each task are sampled sequentially, we have the freedom to decide whether more samples are required at any point. Consider a specific task. At iteration $1 \leq n < N$, we first generate Y_n. Then, an early stopping algorithm $\mathcal{A}\colon \mathcal{Y}^* \to \{\texttt{stop}, \texttt{continue}\}$ decides whether the sampling continues or not. If $\mathcal{A}(Y_{1:n}) = \texttt{stop}$, we terminate sampling for this task and return $S(Y_{1:n})$. If the algorithm never decides to stop, all N samples are generated and $S(Y_{1:N})$ is returned. By convention, we define $\mathcal{A}(y_{1:N}) = \texttt{stop}$ for any N outputs $y_{1:N}$. Let the cost of each sample be λ, measured in accuracy units, and $\tau_{\mathcal{A}}$ be the stopping time for this process:

$$\tau_{\mathcal{A}} = \min\{1 \leq n \leq N \;\mid\; \mathcal{A}(Y_{1:n}) = \texttt{stop}\}.$$

The expected utility of $\mathcal{A}$ is defined as $\mathbb{P}\big(S(Y_{1:\tau_{\mathcal{A}}}) = z^*\big) - \lambda\tau_{\mathcal{A}}$. In other words, it is the accuracy of the process minus the cost of samples.

4 Fixed-lookahead Stability for Early Stopping

Fix $1 \leq n < N$. The optimal decision to stop or continue sampling at iteration n depends on whether, given $Y_{1:n}$, the increase in the accuracy dominates the cost of further sampling. Formally, we should stop if

$$\mathbb{P}\big(S(Y_{1:\tau}) = z^* \mid Y_{1:n}\big) - \mathbf{1}\big[S(Y_{1:n}) = z^*\big] \leq \lambda\mathbb{E}\big[\tau - n \mid Y_{1:n}\big]. \tag{1}$$

Here, τ is the stopping time if we decide to continue sampling at iteration n and follow the optimal early-stopping algorithm afterward. The lefthand side is the conditional increase in accuracy of the process if we continue sampling, and the righthand side is expected cost of the $\tau - n$ additional samples. There are two main challenges in detecting whether condition (1) holds. First, the stopping time τ is stochastic, and its dynamics depends both on the LLM's output distribution p and the decisions made in the future. Second, no information about the correct answer z^* is given to us. We aim to detect a surrogate condition that avoids these complexities and reach a more easily verifiable condition.

The stopping time τ takes a random value from $n+1$ to N. We simplify the condition by replacing it with $n + M$ for a fixed parameter $M > 0$. With this change, we need to compare the increase in accuracy achieved by M new samples with the cost of these samples. Specifically, we stop if

$$\mathbb{P}\big(S(Y_{1:n+M}) = z^* \mid Y_{1:n}\big) - \mathbf{1}\big[S(Y_{1:n}) = z^*\big] \leq \lambda M. \tag{2}$$

The remaining challenge is the dependence on z^* that is unknown. We propose replacing the conditional accuracy-gain term with an upper bound for it. This will give us a more conservative stopping rule: if the upper bound is less than λM, the accuracy gain is also less than λM. We note that

$$\mathbb{P}\big(S(Y_{1:n+M}) = z^* \mid Y_{1:n}\big) - \mathbf{1}\big[S(Y_{1:n}) = z^*\big] \leq \mathbb{P}\big(S(Y_{1:n+M}) \neq S(Y_{1:n}) \mid Y_{1:n}\big).$$

Intuitively, the increase in the accuracy is upper bounded by the conditional probability that the new M samples change the selector's decision. By replacing this upper bound into (2), defining $\rho \triangleq 1 - \lambda M$, and simple arithmetics, we arrive at the stopping condition

$$P_n \triangleq \mathbb{P}\big(S(Y_{1:n+M}) = S(Y_{1:n}) \mid Y_{1:n}\big) \geq \rho. \tag{3}$$

We refer to the conditional probability above as the *stability probability* and denote it by P_n. We can rewrite the stopping condition as $P_n \geq \rho$. Note that the parameters ρ and M are knobs that control the accuracy-cost trade-off by changing how easily the stopping condition is triggered. In common selectors that we are interested in, it is more likely that $S(Y_{1:n+M})$ is the same as $S(Y_{1:n})$ if M is set to a smaller value. Therefore, a smaller M or ρ leads to a looser condition that stops sampling more aggressively to save more cost. In the next section, we show how we can obtain a lower bound for the stability probability P_n. If the lower bound is larger than ρ, we will infer the stopping condition holds and stop sampling.

5 Bounding Stability Probability for Union-consitent Selectors

For any set of outputs $y_{1:k}$ and integer $t > 0$, define the distribution of the selector's choice over the *prefix* outputs $y_{1:k}$ plus t new samples as

$$q_t(z; y_{1:k}) \triangleq \mathbb{P}\big(S(y_{1:k} \cup Y'_{1:t}) = z\big) \qquad (z \in \mathcal{Z}),$$

where $Y'_1, \dots, Y'_t$ are independently sampled from p. This is a generalization of the stability probability that we want to bound since we have $P_n = q_M(S(Y_{1:n}); Y_{1:n})$. If we have access to fresh independent samples, estimating and bounding $q_t(z; y_{1:k})$ is simple. Given a *subsampling pool* of independent samples $Y'_1, \dots, Y'_T$ for some $T \geq t$, we can estimate $q_t(z; y_{1:k})$ by the following U-statistic estimator:

$$\hat{q}_t(z; y_{1:k}, Y'_{1:T}) \triangleq \binom{T}{t}^{-1} \sum_{I \subseteq [T], |I| = t} \mathbf{1}\big[S(y_{1:k} \cup Y'_I) = z\big] \qquad (z \in \mathcal{Z}). \tag{4}$$

That is, we iterate over all subsets Y'_I of $Y'_{1:T}$ with size t, evaluate the selector's choice $S(y_{1:k} \cup Y'_I)$ for each of them, and use the fraction of subsets that yield z as the estimate for its probability. In cases that numerating all index subsets I is computationally too expensive, a large enough number random subsets can be used instead. We can also obtain a high-probability lower bound by the Hoeffding's inequality for U-statistics (Hoeffding, 1963, Equation 5.7). With probability of at least $1 - \delta$, for all $z \in \mathcal{Z}$, we have

$$q_t(z; y_{1:k}) \geq \hat{q}_t(z; y_{1:k}, Y'_{1:T}) - \sqrt{\log(|\mathcal{Z}|/\delta)/2\lfloor T/t \rfloor} \triangleq q^*_{t,\delta}(z; y_{1:k}, Y'_{1:T}), \tag{5}$$

where we let $q^*_{t,\delta}(z; y_{1:k}, Y'_{1:T})$ denote the lower bound.

There is a fundamental challenge in directly using the same approach to estimate or bound the stability probability, which is equal to $q_M(S(Y_{1:n}); Y_{1:n})$. The given samples $Y_{1:n}$ at iteration n need to be used in the place of the prefix $y_{1:k}$, and we don't have access to any other samples to use as the subsampling pool $Y'_{1:T}$. We overcome this issue by utilizing a natural property that the selectors of our interest from Section 3 have, and even more generally, is desirable for any selector used in repeated sampling.

Definition 1 (Union Consistency)**.** Let $y^{(1)}_{1:n_1}$ and $y^{(2)}_{1:n_2}$ be two arbitrary disjoint sets of outputs. Selector S is union-consistent if $S(y^{(1)}_{1:n_1}) = S(y^{(2)}_{1:n_2}) = z$ implies $S(y^{(1)}_{1:n_1} \cup y^{(2)}_{1:n_2}) = z$.

Intuitively, if a union-consistent selector picks a final answer z given each of two disjoint sets of outputs, it also picks z given their union. This condition resembles the consistency condition in social choice, where S is the voting rule and the outputs are the voters (Young, 1974). For the rest of this paper, we assume S is union-consistent. It can be verified that the popular choices S_{BoN}, S_{SC}, and S_{WBoN} introduced in Section 3 satisfy this property. Union-consistency allows us to reason about the selector's choice with $n + M$ outputs by partitioning the outputs into disjoint subsets and reasoning about each subset individually. We introduce two partitioning schemes in the following sections that are useful depending on how large n is compared to M.

5.1 Prefix and Subsampling Pool Partitioning

As we discussed, we cannot estimate and bound the stability probability $q_M(S(Y_{1:n}); Y_{1:n})$ directly because all of $Y_{1:n}$ are used as the prefix outputs of this term and no samples are left for the estimation. Here, we show that we can instead use only a subset of $Y_{1:n}$ as the prefix of the estimated probability and use union-consistency to derive a lower bound for the stability probability.

Assume $M \leq n$ and b_n is chosen such that $M \leq b_n \leq n$. The choice of b_n is specified in Section 6, when we introduce the algorithm. Define Ω to be the outcome space and define event $E_1(n)$ as

$$E_1(n) \triangleq \{\omega \in \Omega \mid S(Y_{1:b_n}) = S(Y_{1:n}) \text{ and } S(Y_{b_n+1:n+M}) = S(Y_{1:n})\}. \tag{6}$$

This is the event that selector's choice on both $Y_{1:b_n}$ and $Y_{b_n+1:n+M}$ equal $S(Y_{1:n})$. Since $Y_{1:b_n}$ and $Y_{b_n+1:n+M}$ partition $Y_{1:n+M}$, by union-consistency of S, event $E_1(n)$ implies $S(Y_{1:n+M}) = S(Y_{1:n})$. Therefore, $\mathbb{P}(E_1(n) \mid Y_{1:n})$ is a lower bound for the stability probability. Now, note that

$$P_n \geq \mathbb{P}(E_1(n) \mid Y_{1:n}) = \mathbf{1}\big[S(Y_{1:b_n}) = S(Y_{1:n})\big] \cdot q_M(S(Y_{1:n}); Y_{b_n+1:n}).$$

Unlike $q_M(z; Y_{1:n})$, we can estimate $q_M(z; Y_{b_n+1:n})$ that has appeared here for any z. Out of $Y_{1:n}$, the outputs $Y_{b_n+1:n}$ are used as the prefix and the rest of available outputs, that is $Y_{1:b_n}$, can be used as the subsampling pool. This gives an estimation of $\mathbb{P}(E_1(n) \mid Y_{1:n})$ as

$$\mathbb{P}(E_1(n) \mid Y_{1:n}) \approx \mathbf{1}\big[S(Y_{1:b_n}) = S(Y_{1:n})\big] \cdot \hat{q}_M(S(Y_{1:n}); Y_{b_n+1:n}, Y_{1:b_n}) \triangleq \hat{L}_1(n), \tag{7}$$

where we defined $\hat{L}_1(n)$ to be this estimated lower bound. If we use $q^*_{M,\delta}$ instead of $\hat{q}_M$, we get a high-probability lower bound that we denote by $L^*_1(n, \delta)$.

It is insightful to note the dynamics of choosing b_n as it decides the subsampling pool's size b_n and the prefix size $n - b_n$. On the one hand, a larger b_n leads to a more accurate estimation of $q_M(z; Y_{b_n+1:n})$, since according to (5), the error scales by $\sqrt{M/b_n}$. On the other hand, $q_M(z; Y_{b_n+1:n})$ will have a smaller prefix when b_n is large. In most selectors of interest, M samples are more likely to influence the outcome when the prefix is small, and we may underestimate the stability probability.

The approximate and high-probability lower bounds $\hat{L}_1(n)$ and $L^*_1(n, \delta)$ will help us detect the stopping condition $P_n \geq \rho$ when n is sufficiently large. However, their utility is limited when n is small. Since the size of the subsampling pool b_n has to be larger than M and smaller than the number of available samples n, this approach is not possible if $n < M$. Even if $n \geq M$ but n is small, the estimation procedure may be too inaccurate. Based on (5), the error of our estimation procedure scales as $\sqrt{M/b_n}$. Even if we allocate all samples to the subsampling pool and set $b_n = n$, this error can be prohibitively too large for large M and small n. To address this issue, we introduce another partitioning scheme.

5.2 Future Samples Partitioning

The error of estimating $q_t(z; y_{1:k})$ with a subsampling pool of size T, as shown in (5), scales as $\sqrt{t/T}$. To enable accurate estimation even in small n, we introduce another partitioning such that t in our estimation can be made small and all samples are allocated to the subsampling pool. We partition the M samples $Y_{n+1:n+M}$ into k_n (almost) equal-sized blocks, where the choice of k_n will be discussed in Section 6. Let $m_n = \lfloor M/k_n \rfloor$ and $r_n = M - k_n m_n$. Then, $\{n+1, \ldots, n+M\}$ can be partitioned into $B_1, \ldots, B_{k_n}$ such that $|B_i| = m_n + 1$ for $i \leq r_n$ and $|B_i| = m_n$ for $i > r_n$. Then, we define event $E_2(n)$ as

$$E_2(n) \triangleq \{\omega \in \Omega \mid \forall i \in [k_n] \colon S(Y_{B_i}) = S(Y_{1:n})\}.$$

Since the sets Y_{B_i} partition $Y_{n+1:n+M}$, by union-consistency of S, event $E_2(n)$ implies $S(Y_{n+1:n+M}) = S(Y_{1:n})$, which itself implies $S(Y_{1:n+M}) = S(Y_{1:n})$. Consequently, we have

$$P_n \geq \mathbb{P}(E_2(n) \mid Y_{1:n}) = q_{m_n+1}(S(Y_{1:n}); \emptyset)^{r_n} q_{m_n}(S(Y_{1:n}); \emptyset)^{k_n - r_n}.$$

We can estimate $q_{m_n}(z; \emptyset)$ and $q_{m_n+1}(z; \emptyset)$ for any z by using $Y_{1:n}$ as the subsampling pool. This gives us the estimate $\hat{L}_2(n)$ defined by

$$\mathbb{P}(E_2(n) \mid Y_{1:n}) \approx \hat{q}_{m_n+1}(S(Y_{1:n}); \emptyset; Y_{1:n})^{r_n} \hat{q}_{m_n}(S(Y_{1:n}); \emptyset; Y_{1:n})^{k_n - r_n} \triangleq \hat{L}_2(n). \tag{8}$$

If $\hat{q}_{m_n}$ and $\hat{q}_{m_n+1}$ are replaced with q^*_{δ,m_n} and q^*_{δ,m_n+1}, we get a high-probability lower bound denoted by $L^*_2(n, \delta)$.

We have used all of our samples for subsampling pool here to allow a reasonable estimate even with small n. For this estimation to be possible, we require $m_n < n$. Also, the estimation error in this lower bound scales by $m_n/n \approx M/(nk_n)$. Considering $m_n = \lfloor M/k_n \rfloor$, both of these constraints can be ensured by choosing a large enough k_n. It is also worth mentioning that larger k_n and smaller m_n mean a stricter condition for $E_2(n)$ and widen the gap between $\mathbb{P}(E_2(n) \mid Y_{1:n})$, which is estimated, and the stability probability.

6 SANE Algorithm

The lower bounds for the stability probability derived in Section 5 form the basis of SANE. Starting from $n = n_{\min}$, we use the high-probability lower bound from Section 5.2 and stop if $L^*_2(n, \delta/N) \geq$

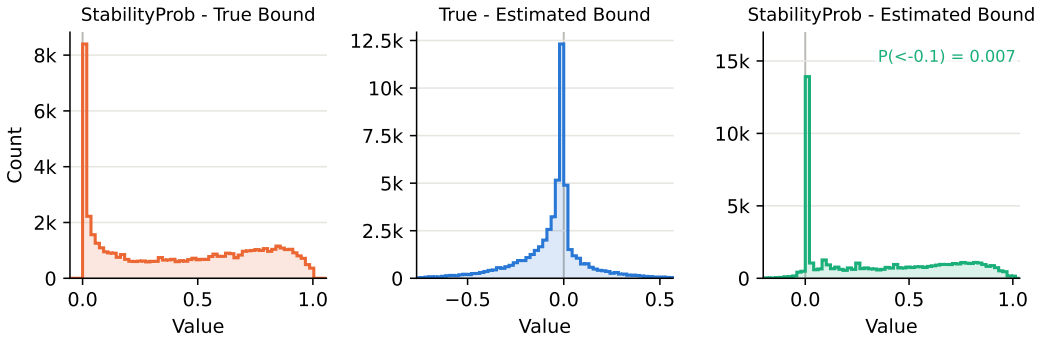


Figure 2: Histogram of the difference between the stability probability and true lower bound *(Left)*, true and estimated lower bounds *(Middle)*, and the stability probability and estimated lower bound *(Right)*. Despite estimation error, the estimated lower bound is almost always valid.

ρ for some $\delta > 0$. The main design decision here is the choice of k_n that decides $m_n = \lfloor M/k_n \rfloor$. To make $\mathbb{P}(E_2(n) \mid Y_{1:n})$ an increasingly tighter lower bound, we want m_n to grow as n increases. At the same time, we want the estimation error $\sqrt{m_n/n}$ to go to zero. These are classic conditions in m-out-of-n bootstrapping and subsampling (Bickel & Sakov, 2008). We follow the standard choice of $m_n \asymp n^\alpha$ for some hyperparameter $\alpha \in (0, 1)$, which leads to estimation error $\mathcal{O}(n^{(\alpha-1)/2})$. This is achieved by setting $k_n = \max\left(1, \left\lfloor M/\lfloor n^\alpha \rfloor \right\rfloor\right)$.

Once $\lfloor n^\alpha \rfloor > M/2$, the number of blocks k_n with this rule reaches 1. This means that we have enough samples to estimate q_M without partitioning the M future samples. At this point, we start using the lower bound from Section 5.1 and stop if $L_1^*(n, \delta/N) \geq \rho$. When deriving this lower bound, we need b_n to increase as n grows to reduce the estimation error proportional to $\sqrt{M/b_n}$. Also, b_n should not be too close to n so that the prefix size $n - b_n$ also grows. We choose to keep the rate of estimation error $\mathcal{O}(n^{(\alpha-1)/2})$. This choice leads to $b_n = \min\left(n, \lfloor Mn^{1-\alpha} \rfloor\right)$.

Lower bounds $L_1^*(n, \delta/N)$ and $L_2^*(n, \delta/N)$ are smaller than the estimated values for $\mathbb{P}\big(E_1(n) \mid Y_{1:n}\big)$ and $\mathbb{P}\big(E_2(n) \mid Y_{1:n}\big)$ to account for any estimation error. Using them instead of the estimated values $\hat{L}_1(n)$ and $\hat{L}_2(n)$ gives us a high-probability guarantee. With probability at least $1 - \delta$, at the time that SANEstops sampling, the condition in (3) holds. The downside of aiming for this theoretical guarantee is that the algorithm becomes too pessimistic about the quality of its estimates and require many samples to theoretically guarantee stability. For example, it requires $b_n > 6600$ to bound the estimation error by 0.1 when $|Z| = 3$, $\delta = 0.05$, $N = 128$, and $M = 16$.

In practice, the estimation error is much better than the worst-case. We observe that it is extremely rare to overestimate the bound such that the final result becomes larger than the stability probability. We investigate this empirically on MMLU-Pro-Math (Wang et al., 2024b) and 21 combinations of LLM, reward model, and selector. Using 512 outputs for each prompt, we find the true value of stability probability and the lower bound we aim to estimate. In Figure 2, for $n = 16$, $\alpha = 0.5$, and $M = 16$, we show the histogram of the gap between the lower bound and the stability probability, the estimation error, and the end-to-end difference between the estimated bound and the stability probability. We observe that overestimation of the bound very rarely dominates the looseness of the bound. The final estimate overshoots the stability probability by more than 0.1 in less than 0.7% of trials. For the smaller $n = 4$ this figure only raises to 5.3%. To achieve empirical efficiency at the cost of theoretical guarantees, we set $n_{\min} = 4$ in all our experiments and use approximate lower bounds $\hat{L}_1(n)$ and $\hat{L}_2(n)$ in practice. In the estimator (4), we iterate over 2000 random subsets and shuffle outputs at every iteration.

7 Empirical Results

Setup. We use the dataset of Rakhsha et al. (2025) for LLM generations. This dataset includes 500 randomly selected prompts from each of the five benchmarks: MATH500 (Lightman et al.,

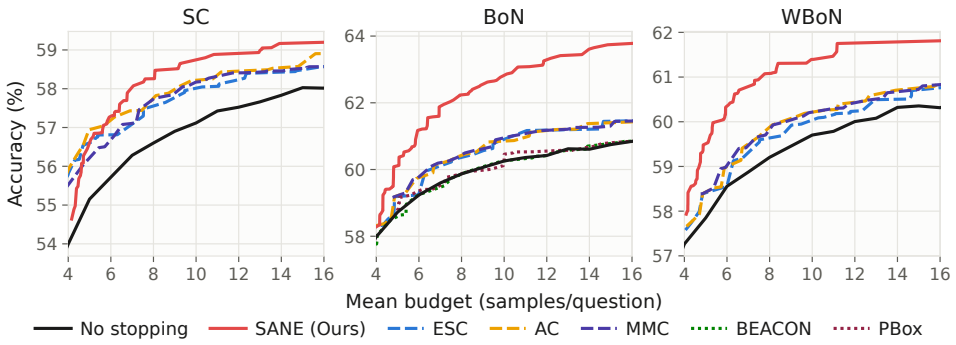


Figure 3: Comparison of early-stopping algorithms with tuned parameters at every mean budget.

2023), GSM8K (Cobbe et al., 2021), MMLU-Pro (Wang et al., 2024b) questions in math (MMLU-Pro-Math) and chemistry (MMLU-Pro-Chem), and Common Sense QA (CSQA) (Talmor et al., 2018). For each prompt, 512 outputs are generated with chain-of-thought prompting by each of three LLMs and each of the outputs are scored by three reward models. We refer to Rakhsha et al. (2025) for more details on the prompts and the sampling configuration. To keep the number of setups we present managable, we combine the data points of all nine LLM-reward combinations for each benchmark.

Baselines Comparisons. We pick three baseline algorithms specialized for self-consistency: Early-stopping Self-consistency (ESC) (Li et al., 2024), which stops once a fixed-window of constant answers is observed, Martingale Majority Certificate (MMC) (Cordero-Encinar & Duncan, 2025) which uses martingale certificates, and among approaches based on Bayesian inference, Adaptive-Consistency (AC) (Aggarwal et al., 2023). We extend the SC-specialized algorithms with BoN and WBoN by following the same stopping rule but returning the selector's choice instead of the majority answer. We also compare against BoN-specialized algorithms Pandora Box (PBox) (Kalayci et al., 2025), which fits a shifted exponential distribution to the reward tails, and BEACON (Wan et al., 2025b) that uses normal distributions. Lastly, we add two standard heuristics that are common for detecting convergence of sequences. Persistence (Pers) stops sampling once the selector's choice has been constant for a fixed number of iterations. Two-scale Agreement (TSA) stops if the selector's choice with all n outputs matches the choice with the first βn outputs for some chosen $\beta \in (0, 1)$.

With all early-stopping algorithms, we have parameters that control how aggressively the sampling is stopped to save cost. For SANE, the parameters N, ρ and M are the knobs where smaller values lead to lower cost, and α is a hyperparameter. For a fair comparison, we run all algorithms with a grid sweep over all their parameters and hyperparameters. The details of these sweeps are given in the supplementary material. In Figure 3, we compare the algorithms on MMLU-Pro-Math across all three selectors. At each value of mean budget, we plot the highest accuracy possible in each algorithm's parameter sweep. The results show that SANE forms the pareto frontier. The gap is larger in BoN and WBoN, where parametric modeling of the process is harder. Surprisingly, the BoN-specialized algorithms have no advantage over fixed-budget sampling. We hypothesize that this is due to violation of their modeling assumptions that are better motivated and tested for alignment tasks instead of our reasoning ones. The equivalent figures for other benchmarks are given in the supplementary material.

In another experiment, we equally split the data points of each benchmark to validation and test sets. In each setup-selector pair, we use the validation set to pick the parameters of each algorithm that reach the accuracy of fixed 16-output budget (plus a 0.1 margin) with the smallest mean cost. Table 1 shows how much cost each algorithm saves with its tuned parameters on the test set. In all cases, SANE beats all baselines and provides 42.4%–73.9% reduction in cost.

Parameter Choice for SANE. We investigate how each the parameters N, ρ, M, and α impact the cost and accuracy of SANE. Figure 4 shows the accuracy and mean cost of SANE on MMLU-Pro-Math when varying each of the parameters from the default choice of $N = 32$, $\rho = 0.5$, $M = 8$, and $\alpha = 0.5$. We observe that all configurations of SANE fall above the fixed-budget curve. This shows that even though these choices impact the accuracy-cost trade-off, the advantage over fixed-budget

Table 1: Budget saved against the base algorithm at $N = 16$, at equal accuracy (%). N/A means the accuracy could not be reached.

Base	Benchmark	SANE (Ours)	ESC	AC	MMC	BEACON	PBox	Pers	TSA
SC	CSQA	**72.1**	63.4	64.2	59.4	–	–	57.2	49.4
	GSM8K	**69.6**	52.3	54.7	61.2	–	–	42.8	48.9
	MATH500	**42.4**	25.7	27.9	26.2	–	–	24.0	13.8
	MMLU-Pro Chem	**45.2**	27.3	24.1	27.3	–	–	22.7	18.8
	MMLU-Pro Math	**61.0**	32.0	43.9	43.6	–	–	35.5	46.0
BoN	CSQA	**63.2**	35.3	35.3	30.5	−11.7	−8.4	19.6	N/A
	GSM8K	**71.5**	62.6	61.7	65.5	−11.6	−4.8	28.5	54.8
	MATH500	**63.4**	45.2	46.1	49.6	−5.5	6.3	35.0	43.8
	MMLU-Pro Chem	**57.5**	3.2	9.8	11.2	−19.1	−5.0	33.2	28.7
	MMLU-Pro Math	**63.6**	33.2	28.8	28.6	−5.1	−12.5	37.7	41.8
WBoN	CSQA	**73.9**	64.0	N/A	N/A	–	–	58.5	49.5
	GSM8K	**71.4**	52.3	54.7	52.2	–	–	19.2	N/A
	MATH500	**47.4**	18.2	31.8	25.2	–	–	18.6	N/A
	MMLU-Pro Chem	**52.0**	3.2	9.8	7.7	–	–	27.2	14.2
	MMLU-Pro Math	**63.8**	20.7	27.2	28.6	–	–	35.7	46.2

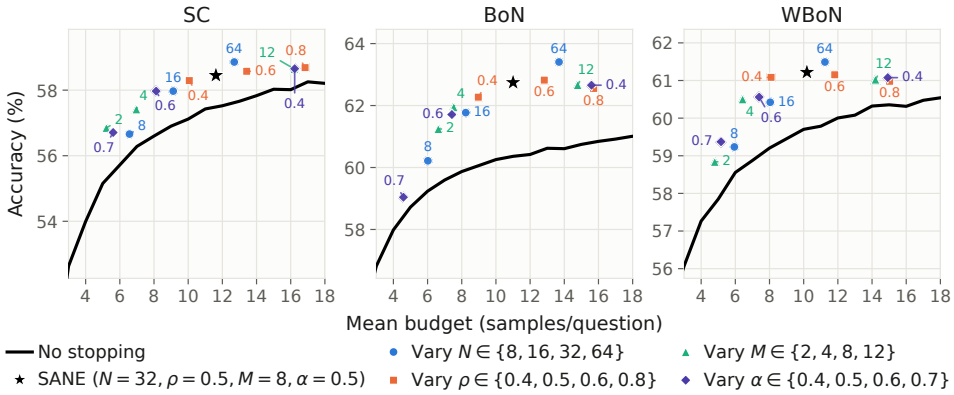


Figure 4: Impact of varying each parameter of SANE on its accuracy and cost on MMLU-Pro-Math.

sampling is persistent. The figure shows that smaller value of ρ, M, N and larger values of α leads to lower cost.

8 Conclusion

In this paper, we introduced SANE, an early-stopping algorithm for repeated sampling from LLMs that requires only black-box access to the selection mechanism and no training. We identified fixed-lookahead stability as a surrogate stopping rule for early-stopping. This rule suggests to stop sampling once the probability that the final answer remains the same after M more samples is larger than a threshold. We show that this probability can be bounded under a mild union-consistency assumption on the selector. The bound is derived by partitioning samples into blocks, and using subsampling to estimate the distribution of the selected answer from each block. The conservative version of SANE holds a high-probability guarantee to only trigger when the stability condition is true. Empirically, we evaluate SANE on five benchmarks and three selection algorithms with the data of three LLMs and three reward models. Across all setups, SANE outperforms the existing early-stopping algorithms. The advantage is more pronounced in reward-guided selectors whose parametric modeling is more difficult.

AI USE STATEMENT

In this work, we heavily used AI tools for the implementation of the algorithms and experiments. The code was evaluated at a high level and in critical segments. These tools were used lightly for sentence-level polishing of the text.

REFERENCES

Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with llms. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12375–12396. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.761. URL https://aclanthology.org/2023.emnlp-main.761/. 1, 3, 8

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*, pp. 967–985, 2008. 7

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. 1

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, 2021. 8

Paula Cordero-Encinar and Andrew B. Duncan. Certified self-consistency: Statistical guarantees and test-time training for reliable reasoning in llms, 2025. 1, 3, 8

Mehul Damani, Idan Shenfeld, Andi Peng, Andreea Bobu, and Jacob Andreas. Learning how hard to think: Input-adaptive allocation of lm computation, 2024. 3

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. 13

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. ISSN 0162-1459. doi: 10.2307/2282952. 5

Audrey Huang, Adam Block, Qinghua Liu, Nan Jiang, Akshay Krishnamurthy, and Dylan J. Foster. Is best-of-n the best of them? coverage, scaling, and optimality in inference-time alignment. In *Forty-second International Conference on Machine Learning*, 2025a. URL https://openreview.net/forum?id=QnjfkhrbYK. 1

Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiaxin Huang. Cats: Calibrated test-time scaling for efficient llm reasoning. In *The Fourteenth International Conference on Learning Representations*, 2025b. 3

Jingkai Huang, Will Ma, and Zhengyuan Zhou. Optimal bayesian stopping for efficient inference of consistent llm answers, 2026. 1, 3

Yuki Ichihara, Yuu Jinnai, Tetsuro Morimura, Kenshi Abe, Kaito Ariu, Mitsuki Sakamoto, and Eiji Uchibe. Evaluation of best-of-n sampling strategies for language model alignment. *Transactions on Machine Learning Research*, 2025. 1

Yuu Jinnai, Tetsuro Morimura, Kaito Ariu, and Kenshi Abe. Regularized best-of-n sampling to mitigate reward hacking for language model alignment. In *ICML 2024 Workshop on Models of Human Feedback for AI Alignment*, 2024. 1

Yusuf Kalayci, Vinod Raman, and Shaddin Dughmi. Optimal stopping vs best-of-n for inference time optimization, 2025. 1, 3, 8

Junpei Komiyama, Daisuke Oba, and Masafumi Oyamada. Best-of-infinity: Asymptotic performance of test-time llm ensembling. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=3qiCnLf3jf. 1, 3

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. 1

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning, 2024. 1, 2, 8

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. 1, 7

Chris Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, Yang Liu, and Yahui Zhou. Skywork-reward-v2: Scaling preference data curation via human-ai synergy. *arXiv preprint arXiv:2507.01352*, 2025. 13

OpenAI. Learning to reason with llms. https://openai.com/index/learning-to-reason-with-llms/, 2024. Accessed: 2025-05-10. 1

Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/. 13

Amin Rakhsha, Kanika Madan, Tianyu Zhang, Amir massoud Farahmand, and Amir Khasahmadi. Majority of the bests: Improving best-of-n via bootstrapping. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=aEAbRPXV37. 1, 7, 8, 13

Vinod Raman, Hilal Asi, and Satyen Kale. Adabon: Adaptive best-of-n alignment, 2026. 3

Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. From words to watts: Benchmarking the energy costs of large language model inference. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–9, 2023. doi: 10.1109/HPEC58863.2023.10363447. 1

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. 1

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*, 2018. 8

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. 13

Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. Reasoning aware self-consistency: Leveraging reasoning paths for efficient llm sampling, 2025a. 3

Guangya Wan, Zixin Stephen Xu, Sasa Zorc, Manel Baucells, Mengxuan Hu, Hao Wang, and Sheng Li. Beacon: Bayesian optimal stopping for efficient llm sampling, 2025b. 1, 3, 8

Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. Interpretable preferences via multi-objective reward modeling and mixture-of-experts. In *Conference on Empirical Methods in Natural Language Processing*, 2024a. 13

Xinglin Wang, Shaoxiong Feng, Yiwei Li, Peiwen Yuan, Yueqi Zhang, Chuyi Tan, Boyuan Pan, Yao Hu, and Kan Li. Make every penny count: Difficulty-adaptive self-consistency for cost-efficient reasoning. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 6919–6932. Association for Computational Linguistics, 2025. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.383. 3

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. 1

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. 7, 8

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. 13

H. P Young. An axiomatization of borda's rule. *Journal of Economic Theory*, 9(1):43–52, 1974. ISSN 0022-0531. doi: 10.1016/0022-0531(74)90073-8. 5

Wenting Zhao, Pranjal Aggarwal, Swarnadeep Saha, Asli Celikyilmaz, Jason Weston, and Ilia Kulikov. The majority is not always right: Rl training for solution aggregation. *arXiv preprint arXiv:2509.06870*, 2025. 1

Table 2: Hyperparameter grids. Every grid is also crossed with the 23 values of N; the last column counts settings per setup, N included.

Rule	Base	Hyperparameters	Settings
SANE (Ours)	all	$\rho \in \{0.4, 0.5, \ldots, 0.9\}$ $M \in \{2, 4, \ldots, 16\}$ $\alpha \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$	5,520
ESC	SC	window $w \in \{2, 3, 4, 5, 6, 8, 10, 12, 16\}$ final vote over the stopping window, or over all outputs	414
AC	SC	threshold $\in \{0.7, 0.8, 0.9, 0.95, 0.98, 0.99, 0.995, 0.999\}$	184
MMC	SC	$\varepsilon \in \{0.01, 0.05, 0.1, 0.2, 0.4, 0.6, 0.8, 0.9, 0.95, 0.99\}$ prior: point (B.1), point (B.2), Beta$(\frac{1}{2}, \frac{1}{2})$, Beta$(1, 1)$	920
ESC	BoN, WBoN	window $w \in \{2, 3, 4, 5, 6, 8, 10, 12, 16\}$	207
AC	BoN, WBoN	threshold $\in \{0.7, 0.8, 0.9, 0.95, 0.98, 0.99, 0.995, 0.999\}$	184
MMC	BoN, WBoN	$\varepsilon \in \{0.01, 0.05, 0.1, 0.2, 0.4, 0.6, 0.8\}$ prior: point (B.1)	161
BEACON	BoN	cost $c \in \{0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.3, 0.5\}$	230
PBox	BoN	cost variant: $c \in \{10^{-5}, 3{\cdot}10^{-5}, 6{\cdot}10^{-5}, 10^{-4}, 3{\cdot}10^{-4}, 6{\cdot}10^{-4}, 10^{-3}, 0.002, 0.004, 0.008\}$ target variant: $\tau \in \{0.6, 0.7, 0.8, 0.84, 0.88, 0.92, 0.96, 0.98, 1.0\}$ warm-up $\in \{5, 10, 20\}$ samples, in both variants	1,311
Pers	all	patience $\in \{1, 2, 3, 4, 6, 8, 12, 16\}$ $n_{\min} \in \{1, 4, 8\}$	552
TSA	all	$\beta \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$ $n_{\min} \in \{1, 4, 8\}$ subsample redrawn at every step, or the first outputs	690

A Experiment Details

The three LLMs in the dataset by Rakhsha et al. (2025) used in our experiments are Qwen2.5-3B-Instruct (Qwen Team, 2024), Llama-3.1-8B-Instruct (Grattafiori et al., 2024), and Gemma-2-9B-it (Team et al., 2024). The three reward models are ArmoRM (Wang et al., 2024a), GRM (Yang et al., 2024), and Skywork (Liu et al., 2025).[1].

Parameter sweep and tuning protocol. A *setup* is a pair of a base algorithm (SC, BoN or WBoN) and a benchmark, 15 in total. Each setup averages over its cells: the 3 LLMs for SC, and the 3 LLMs $\times$ 3 reward models for BoN and WBoN. Every stopping rule is swept over the Cartesian product of its hyperparameters and the sampling budget

$$N \in \{6, 8, 10, \ldots, 32,\ 40, 48, 56, 64, 72, 80, 88, 96,\ 128\}$$

with 23 values. Each question holds 512 generations, and a trial at budget N draws its outputs without replacement from them, so every estimate rests on $\lfloor 512/N \rfloor$ trials per question; we cap N at 128 so that every setting is averaged over at least four trials per question. In Table 1, for each stopping and setup, the whole grid, N included, is searched on a validation half of the questions and the selected setting is reported on the other half. A setting is kept only if it was run on every cell of its setup. Table 2 lists the grids. In comparison plots such as Figure 3, we tune and report on the whole data.

B Additional Results

SANE is compared to baselines on MATH500 (Figure 5), MMLU-Pro-Chem (Figure 6), GSM8K (Figure 7), and CSQA (Figure 8). The comparison on MMLU-Pro-Math was given in Figure 3. The impact of varying SANE's parameters is shown on MATH500 (Figure 9), MMLU-Pro-Chem (Figure 10), GSM8K (Figure 11), and CSQA (Figure 12). The comparison on MMLU-Pro-Math was given in Figure 4.

[1]The ids on HuggingFace are `RLHFlow/ArmoRM-Llama3-8B-v0.1`, `Ray2333/GRM-Llama3.2-3B-rewardmodel-ft`, and `Skywork/Skywork-Reward-V2-Llama-3.1-8B`

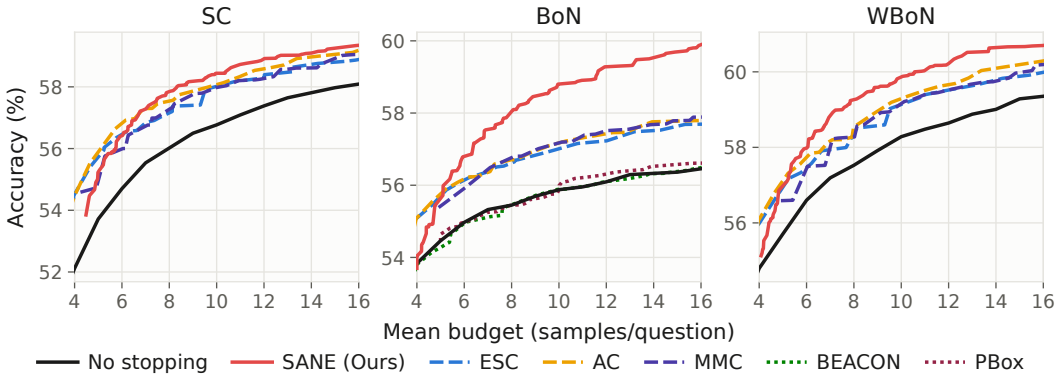


Figure 5: Comparison of early-stopping algorithms with tuned parameters at every mean budget on MATH500.

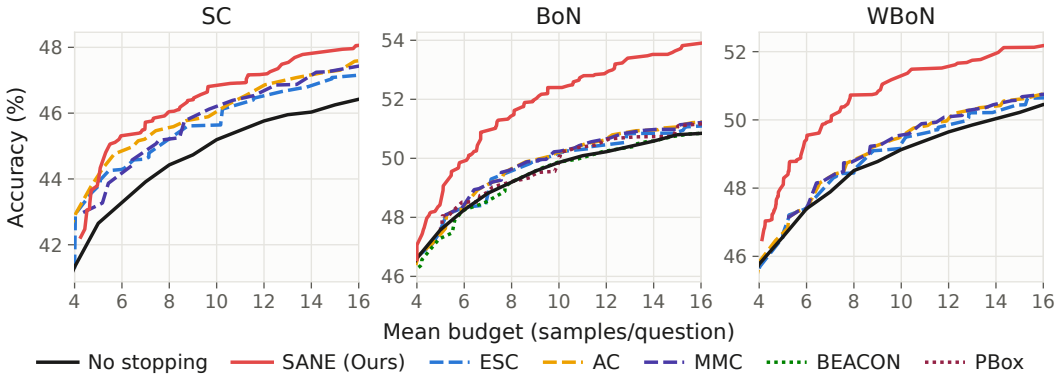


Figure 6: Comparison of early-stopping algorithms with tuned parameters at every mean budget on MMLU-Pro-Chem.

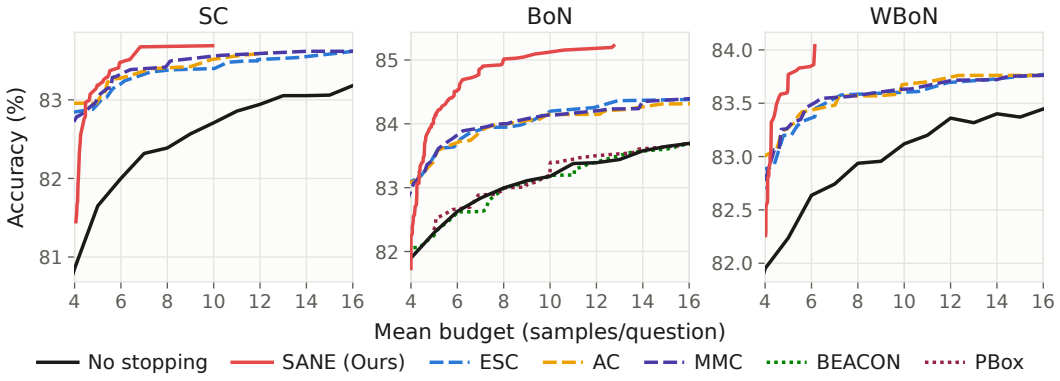


Figure 7: Comparison of early-stopping algorithms with tuned parameters at every mean budget on GSM8K.

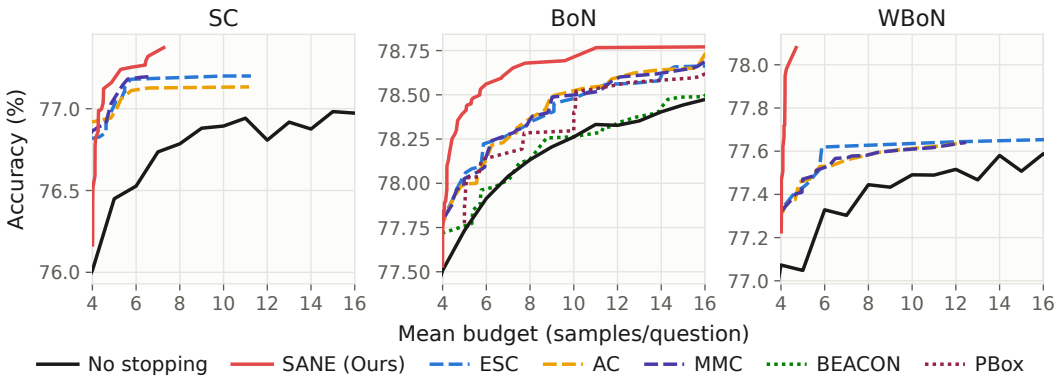


Figure 8: Comparison of early-stopping algorithms with tuned parameters at every mean budget on CSQA.

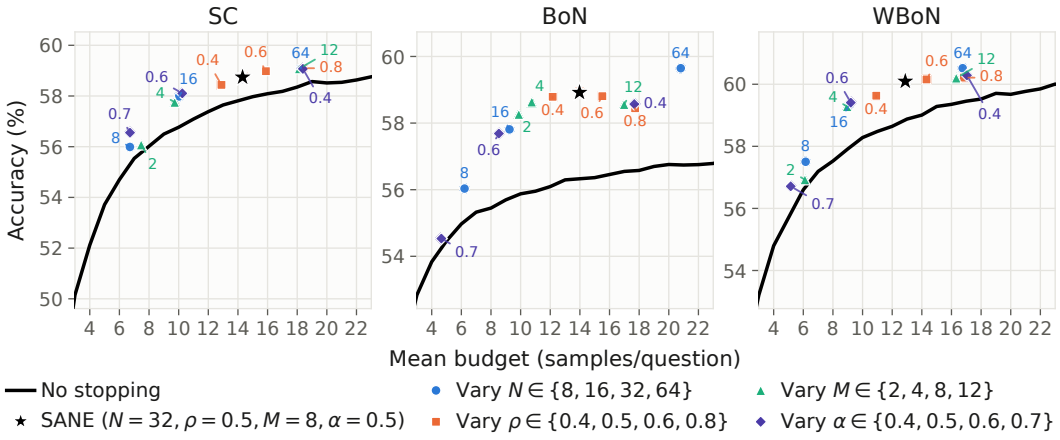


Figure 9: Impact of varying each parameter of SANE on its accuracy and cost on MATH500.

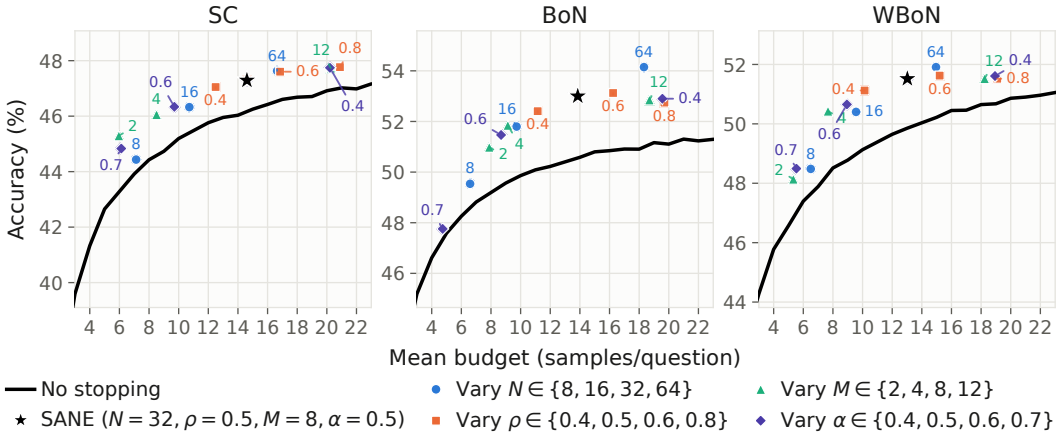


Figure 10: Impact of varying each parameter of SANE on its accuracy and cost on MMLU-Pro-Chem.

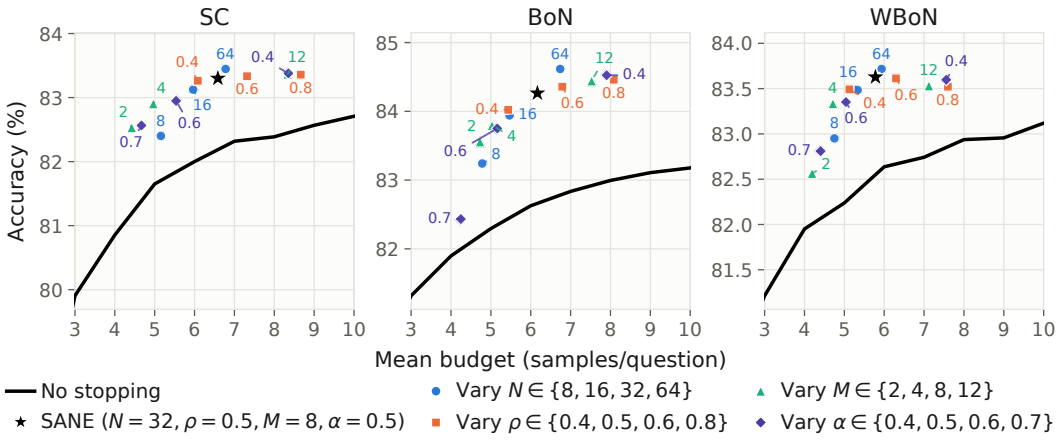


Figure 11: Impact of varying each parameter of SANE on its accuracy and cost on GSM8K.

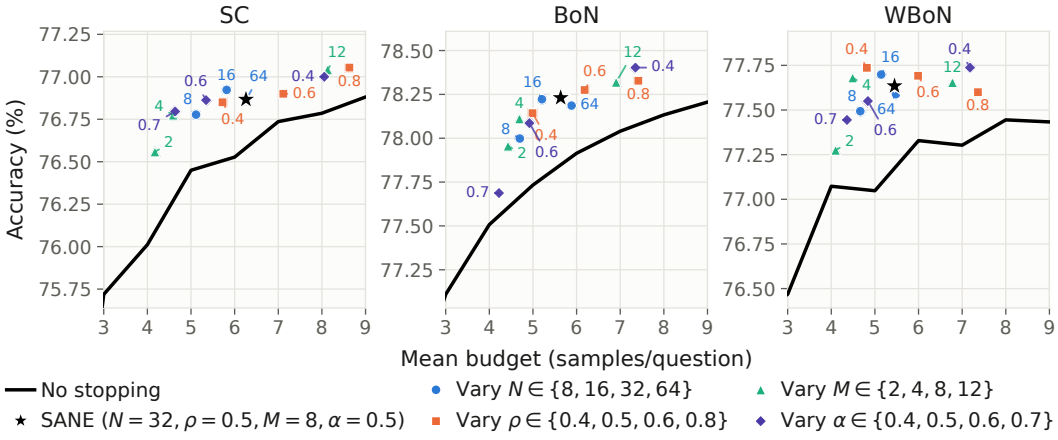


Figure 12: Impact of varying each parameter of SANE on its accuracy and cost on CSQA.