On the Overscaling Curse of Parallel Thinking: System Efficacy Contradicts Sample Efficiency

Yiming Wang
School of Computer Science,
Shanghai Jiao Tong University
yiming.wang@sjtu.edu.cn

Zhuosheng Zhang
School of Computer Science,
Shanghai Jiao Tong University
zhangzs@sjtu.edu.cn

Rui Wang*
School of Computer Science,
Shanghai Jiao Tong University
wangrui12@sjtu.edu.cn

Abstract

Parallel thinking improves LLM reasoning through multi-path sampling and aggregation. In standard evaluations, due to a lack of sample-specific priors, all samples share a global budget chosen to maximize dataset accuracy. However, many samples reach their best accuracy with much smaller budgets, causing low budget utilization. This *contradiction between system efficacy and sample efficiency* constitutes the **Overscaling Curse**. In this paper, we first provide a formal analysis of the overscaling curse and quantify its prevalence and severity in real-world systems. To break it, we propose **Latent Budget Predictor (LATBP)**, which probes model latent representations to predict sample-specific optimal budgets. LATBP significantly improves budget utilization while maintaining dataset accuracy. We further integrate LATBP into the full decoding pipeline, inspiring **Pre-decoding Budget Adaptation (PREADA)**, a paradigm that allocates budgets before decoding to preserve decoding-time parallelization. LATBP substantially improves hardware-aware efficiency in latency and memory, demonstrating both its practical value and the promise of PREADA for efficient parallel decoding.

1 Introduction

Large language models (LLMs) continually update their distributions by large-scale corpora training [45, 4], enabling them to encode broad world knowledge [37, 41] and evolve strong reasoning abilities [17]. As model capacity increases, recent work shows that model distributions contain latent information that is not fully revealed by simple autoregressive decoding [7]. This motivates scaling compute [43, 35] during inference to elicit model potential, termed test-time scaling (TTS).

Among TTS, parallel thinking represents a distinct paradigm that broadens the space of reasoning exploration [12, 68]. This paradigm typically pre-allocates a parallelism budget $N \in \mathbb{N}^+$. During decoding, the model first generates N *independent* paths in parallel via multinomial sampling [20, 21], promoting diversity and broader coverage of the output distribution. Then, an aggregation strategy like majority voting [49] or external verifiers [30] selects the final answer from the N candidates.

In practice, parallel thinking is usually evaluated at the system level: a model-dataset pair $(\pi, \mathcal{D})$ forms a system, and performance is measured by the overall accuracy of model π on dataset $\mathcal{D}$. Due to the lack of sample-specific priors, all samples $\boldsymbol{x} \in \mathcal{D}$ share a global budget $N_{\mathcal{D}}$. Following the

*Corresponding Author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

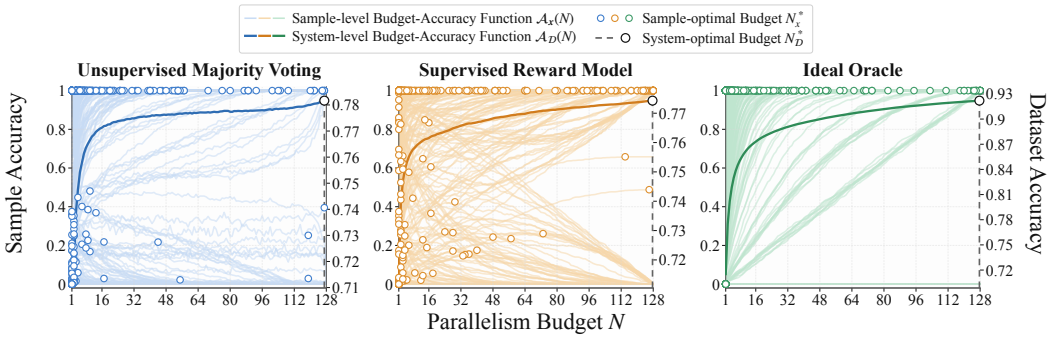


Figure 1: **Illustration of the Overscaling Curse in Parallel Thinking**. This figure illustrates a single system $(\pi, \mathcal{D})$, *i.e.*, evaluating model π on dataset $\mathcal{D}$. Here $\pi =$ `Qwen2.5-7B` and $\mathcal{D} =$ `MATH500`. Each subfigure corresponds to one answer aggregation strategy. The single thick curve shows how overall dataset accuracy varies with the budget, and thin curves show how sample accuracy varies with the budget for each sample $\boldsymbol{x} \in \mathcal{D}$. We term these curves the *"budget-accuracy" functions*. Hollow markers on each curve indicate the minimum budget that achieves the highest accuracy, which we term the *"optimal budget"*. See Section 2 for experimental setups and Section 3.1 for detailed function computation. Illustrations in other systems are shown in Figures 6 – 16 (Appendix A). From this illustration, we can find that: *the system-optimal budget often exceeds the sample-optimal budgets for many individual samples*.

TTS principle of prioritizing performance over cost [43], this budget should first ***prioritize system efficacy***. Thus, the ideal global $N_\mathcal{D}$ is the minimum budget that ***maximizes dataset accuracy***, which we call the system-optimal budget. In contrast, if released from this global constraint, each sample $\boldsymbol{x}$ would have its own ideal budget $N_{\boldsymbol{x}}$, since sampling randomness can induce different expected accuracies under different budgets. Analogously, we call the sample-optimal budget the minimum budget that maximizes the expected accuracy for that sample.

However, when examining real-world systems, *a consistent observation emerges: the system-optimal budget often exceeds the sample-optimal budgets for many individual samples*, as shown in Figure 1 and 6 – 16. This means that, when these samples are forced to follow this global system-optimal budget rather than their own optima, their ***budget utilization decreases***, thereby ***undermining sample efficiency***. This reveals *a fundamental contradiction induced by the global-budget constraint: maximizing system efficacy can come at the cost of sample efficiency*. We refer to this as the **Overscaling Curse**. In this paper, we start from the above phenomenon and provide a formal analysis of the overscaling curse, together with concrete metrics to quantify its severity. Extensive empirical results show that the overscaling curse is both prevalent and severe in real-world systems.

Next, we explore breaking the overscaling curse by improving budget utilization while preserving dataset accuracy. Our foremost step is to relax the global-budget constraint and allow sample-specific allocation. Since each sample has its own optimal budget, we reduce the problem to predicting this sample-optimal budget. As the budget depends on both the sample itself and the model's perception of it, we propose **Latent Budget Predictor (LATBP)**, which probes the model's latent representations of the sample to predict its optimal budget. Empirically, LATBP improves budget utilization by $3 - 6\times$ on average, and a single training can adapt to multiple domains.

Finally, we integrate LATBP into the full parallel decoding pipeline to evaluate its practical hardware-aware efficiency beyond merely reducing the budget size. This integration inspires **Pre-decoding Budget Adaptation (PREADA)**, a new efficient parallel decoding paradigm that allocates budgets before decoding, in contrast to existing efficient methods that adapt budgets during decoding [26, 48, 12]. By avoiding decoding-time blocking, LATBP better preserves full hardware parallelization despite its predictor cost. Empirically, LATBP reduces both latency and memory by over 50% compared to standard parallel decoding, significantly outperforming existing methods, and with less than 2% performance fluctuation, demonstrating its practical value for efficient parallel decoding. More importantly, as the first exploration under PREADA, LATBP demonstrates the feasibility of this paradigm and opens a promising direction for future efficient parallel decoding research[2].

[2]The code repository is available at `https://github.com/Alsace08/overscaling-in-parallel-thinking`

2 Global Experimental Setup

2.1 Implementation of Parallel Thinking

Let $\pi(\boldsymbol{y}|\boldsymbol{x})$ be an autoregressive language model that generates an output sequence $\boldsymbol{y} = (y_1, \ldots, y_{T'})$ conditioned on an input sequence $\boldsymbol{x} = (x_1, \ldots, x_T)$, where each token y_t is sampled sequentially as $y_t \sim \pi(\cdot \mid \boldsymbol{x}, \boldsymbol{y}_{\prec t})$. Parallel thinking consists of two progressive stages [12, 66]: *multi-path sampling* and *answer aggregation*. In the first stage, π independently generates N distinct output sequences $\{\boldsymbol{y}^i\}_{i=1}^N$ for the same $\boldsymbol{x}$, each following the same autoregressive process. In the second stage, an aggregation function $\mathcal{F}$ combines these N sequences to produce the final output $\hat{\boldsymbol{y}} = \mathcal{F}(\boldsymbol{y}^1, \ldots, \boldsymbol{y}^N)$.

Multi-path Sampling. we use multinomial sampling with combined top-k, top-p, and temperature T [11, 21, 20]. Following prior work [60, 17], we set $k = 20$, $p = 0.95$, and $T = 0.6$ by default. Sampling is implemented using the `vLLM` framework with full parallelization on 80G A100 GPUs.

Answer Aggregation. We consider two strategies: *unsupervised majority voting* (self-consistency [49]) and *supervised external verifiers*. For the latter, we use a general-purpose reward model (*Skywork-Reward-V2* [30]) and include an ideal oracle (no verification error) as an upper bound of verifiers. These choices ensure our findings are not tied to specific aggregation methods.

2.2 System Selection: Models and Datasets

In this paper, a system is defined as a model-dataset pair $(\pi, \mathcal{D})$, and the phenomena observed when evaluating π on $\mathcal{D}$ form the meta-object of our study. All systems considered are $\{(\pi, \mathcal{D}) \in \boldsymbol{\pi} \times \boldsymbol{\mathcal{D}}\}$.

Model Set $\boldsymbol{\pi}$. we select four models for our evaluations: `Qwen2.5-7B` [61], `Llama3.1-8B` [15], `Deepseek-R1-Distill-Qwen-7B` [17], and `Qwen3-4B` [60] (the `Instruct-2507` version with `enable_thinking=True`). The first two are non-reasoning models, and the last two are reasoning models. *Together, they provide a broad diversity in model series and reasoning paradigms.*

Dataset Set $\boldsymbol{\mathcal{D}}$. we select seven datasets: `MATH500` [19], `AMC` [32], `AIME24` [33], `AIME25` [34] from the mathematical domain with progressively increasing difficulty; and `GPQA` [38], `MMLU-Pro` [54], `BrowseComp` (en [55] + zh [69]) from diverse domains, including general QA, multi-subject knowledge, and agent tasks. *Together, they provide a broad diversity in difficulty and domain.*

3 Revealing the Overscaling Curse

3.1 Issue Formalization

The overscaling curse stems from *the contradiction between system efficacy and sample efficiency induced by the global-budget constraint*: the former is reflected by *dataset accuracy*, while the latter is reflected by *budget utilization*. We therefore begin by formalizing the **Budget-Accuracy Function** $\mathcal{A}(N)$, which links budget to accuracy at both sample and dataset levels.

We first consider a sample $\boldsymbol{x} \in \mathcal{D}$ without global constraints. Let $\mathcal{A}_{\boldsymbol{x}}(N)$ denote its expected accuracy under budget $N \in [1, N_{\max}]$, where $N_{\max}$ is the largest budget considered[3]. For a fixed N, one run samples N candidates and aggregates them, producing a binary correctness outcome. Since sampling is stochastic, $\mathcal{A}_{\boldsymbol{x}}(N)$ should be estimated as the expected correctness over R repeated N-sampling runs. Naively computing the full function requires repeating this for every $N \in [1, N_{\max}]$, leading to $\mathcal{O}(N_{\max}^2 R)$ sampling cost. To make this tractable, we use subsampling: For $\boldsymbol{x}$, we first generate a reference set of $2N_{\max}$ outputs $\{\boldsymbol{a}_{\boldsymbol{x},n}\}_{n=1}^{2N_{\max}}$. For every N, we uniformly select N outputs without replacement from this set, aggregate them with $\mathcal{F}$, and repeat this operation M times[4], yielding:

$$\begin{aligned} \mathcal{A}_{\boldsymbol{x}}(N) &= \mathbb{E}_{m\sim[M]}\left[\mathbb{I}\left(\mathcal{F}\big(\{\boldsymbol{a}_{\boldsymbol{x},n}^m\}_{n=1}^N\big) = \boldsymbol{y}\right)\right], \\ M &= \min\left\{\tau, \binom{2N_{\max}}{N}\right\}, \qquad \{\boldsymbol{a}_{\boldsymbol{x},n}^m\}_{n=1}^N \sim \mathrm{Uniform}_{\text{w/o repl.}}\left(\{\boldsymbol{a}_{\boldsymbol{x},n}\}_{n=1}^{2N_{\max}}, N\right). \end{aligned} \tag{1}$$

[3]We set $N_{\max} = 128$ in our experiments. During pre-evaluation, we started from $N = 2$ and doubled N until the change in dataset accuracy fell below 0.01, indicating performance convergence. This occurred for most cases by $N = 2^7$.

[4]We set $\tau = 10^5$ in M to avoid unbounded combinatorial enumeration while keeping the Monte Carlo estimation error negligible in practice. With this setting, our stability test yields $p < 0.01$ in all subsequent experiments, supporting the reliability of the estimates. This subsampling procedure requires only one reference generation of size $2N_{\max}$ per sample, reducing the sampling cost to $\mathcal{O}(N_{\max})$ while enabling stable estimation of the full budget-accuracy function.

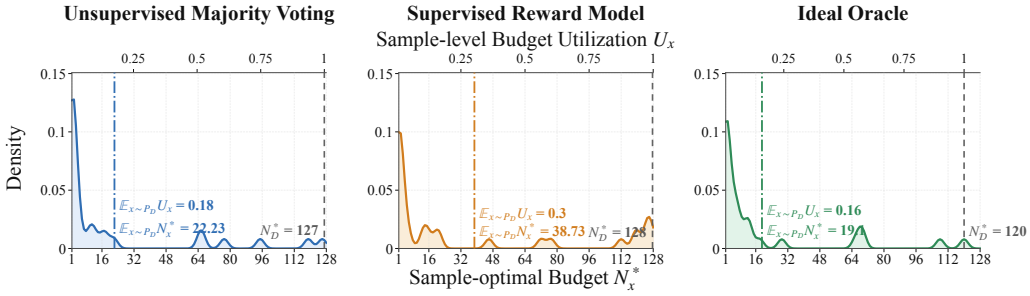


Figure 2: Distribution of *Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding sample-optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_\mathcal{D}^*$. Here $\pi =$ `Qwen3-4B` and $\mathcal{D} =$ `AIME25`. See Figures 17 – 43 in Appendix B for distributions in other systems.

Following the TTS principle of prioritizing performance over cost [43], the ideal budget for $\boldsymbol{x}$ is the minimum budget that achieves its best accuracy. We term it the **Sample-optimal Budget** $N_{\boldsymbol{x}}^*$, *i.e.*, $N_{\boldsymbol{x}}^* = \min(\text{argmax}_{N\in[1,N_{\max}]} \mathcal{A}_{\boldsymbol{x}}(N))$. **(2)** Suppose that $\boldsymbol{x}$ is allocated an actual budget $N_{\boldsymbol{x}}$. Then the ratio between $N_{\boldsymbol{x}}^*$ and $N_{\boldsymbol{x}}$ can be defined as its **Budget Utilization** $\mathcal{U}_{\boldsymbol{x}}$, *i.e.*, $\mathcal{U}_{\boldsymbol{x}} = N_{\boldsymbol{x}}^*/N_{\boldsymbol{x}}$, **(3)** measuring how much of the allocated budget is actually needed by $\boldsymbol{x}$. When $\mathcal{U}_{\boldsymbol{x}} < 1$, the allocated budget is redundant, since any portion beyond $N_{\boldsymbol{x}}^*$ brings no positive accuracy gain for it.

Now, we consider the practical evaluation setting where *sample-specific priors are unavailable, so a global budget is allocated to all samples*. In this case, the dataset-level budget-accuracy function is $\mathcal{A}_\mathcal{D}(N) = \mathbb{E}_{\boldsymbol{x}\sim P_\mathcal{D}}[\mathcal{A}_{\boldsymbol{x}}(N)]$. Analogously, the **System-optimal Budget** is defined as $N_\mathcal{D}^* = \min(\text{argmax}_{N\in[1,N_{\max}]} \mathcal{A}_\mathcal{D}(N))$, and the **Expected Budget Utilization** over $\mathcal{D}$ is $\mathbb{E}_{\boldsymbol{x}\sim P_\mathcal{D}}[\mathcal{U}_{\boldsymbol{x}}]$.

Under this global-budget constraint, *to maximize dataset accuracy*, the global budget should be set to $N_\mathcal{D}^*$. Consequently, each sample's budget utilization becomes $\mathcal{U}_{\boldsymbol{x}} = N_{\boldsymbol{x}}^*/N_\mathcal{D}^*$. However, $N_\mathcal{D}^*$ may exceed the sample-optimal budget of many individual samples. These samples then yield $\mathcal{U}_{\boldsymbol{x}} < 1$, lowering the expected budget utilization $\mathbb{E}_{\boldsymbol{x}\sim P_\mathcal{D}}[\mathcal{U}_{\boldsymbol{x}}]$ and thereby *undermining sample efficiency* over $\mathcal{D}$. Based on this mechanism, ***the essence of the overscaling curse can be formalized as:***

> ***With the global-budget constraint that*** $N_{\boldsymbol{x}} = N_\mathcal{D}^*$ ***for all*** $\boldsymbol{x} \in \mathcal{D}$***, many*** $\boldsymbol{x}$ ***may yield*** $\mathcal{U}_{\boldsymbol{x}} < 1$***, resulting in*** $\mathbb{E}_{\boldsymbol{x}\sim P_\mathcal{D}}[\mathcal{U}_{\boldsymbol{x}}] < 1$.

In summary, the overscaling curse can be viewed as a ***sample–system–sample closed-loop constraint process under the global budget***: (1) Individual samples exhibit heterogeneous $\mathcal{A}_{\boldsymbol{x}}(N)$, each with corresponding $N_{\boldsymbol{x}}^*$; (2) Under the global-budget constraint, these samples are aggregated into $\mathcal{A}_\mathcal{D}(N) = \mathbb{E}_{\boldsymbol{x}\sim P_\mathcal{D}}[\mathcal{A}_{\boldsymbol{x}}(N)]$, jointly determining the system-optimal global budget $N_\mathcal{D}^*$; and (3) $N_\mathcal{D}^*$ is then imposed back on all samples, constraining their $\mathcal{U}_{\boldsymbol{x}}$ and ultimately lowering $\mathbb{E}_{\boldsymbol{x}\sim P_\mathcal{D}}[\mathcal{U}_{\boldsymbol{x}}]$.

3.2 Quantify the Overscaling Curse

We now quantify whether the overscaling curse is prevalent in real-world systems. As formalized in Section 3.1, the closed-loop process ultimately leads to low expected budget utilization $\mathbb{E}_{\boldsymbol{x}\sim P_\mathcal{D}}[\mathcal{U}_{\boldsymbol{x}}]$. Therefore, we use $\mathbb{E}_{\boldsymbol{x}\sim P_\mathcal{D}}[\mathcal{U}_{\boldsymbol{x}}]$ as the primary metric to measure the overscaling degree. A smaller value below 1 indicates more severe overscaling, while values equal to or above 1 suggest that the overscaling curse is absent.

0.0 1.0 >1 0.0 1.0 >1 0.0 1.0 >1

Table 1: *Expected Budget Utilization* $\mathbb{E}_{\boldsymbol{x}\sim P_\mathcal{D}}[\mathcal{U}_{\boldsymbol{x}}]$ across systems $(\pi, \mathcal{D})$ under different aggregation strategies.

Model π \ Dataset $\mathcal{D}$	MATH500	AMC	AIME24	AIME25	GPQA	MMLU-Pro	BrowseComp
Unsupervised Majority Voting							
Qwen2.5-7B	0.09	0.50	0.07	0.16	0.38	0.31	0.23
Llama3.1-8B	0.21	0.24	0.26	0.12	0.24	0.39	0.26
R1-Distill-Qwen-7B	0.20	0.27	0.17	0.22	0.30	0.20	0.36
Qwen3-4B	0.06	0.13	0.34	0.18	0.26	0.31	0.33
Supervised Reward Model							
Qwen2.5-7B	0.17	4.41	1.25	0.41	0.44	0.36	0.27
Llama3.1-8B	0.82	1.11	0.19	0.05	0.35	0.33	0.35
R1-Distill-Qwen-7B	0.35	0.30	0.20	0.26	0.35	0.24	0.39
Qwen3-4B	0.11	4.37	1.42	0.30	0.28	0.45	0.32
Ideal Oracle							
Qwen2.5-7B	0.10	0.27	0.28	0.18	0.33	0.27	0.18
Llama3.1-8B	0.22	0.37	0.33	0.32	0.27	0.25	0.27
R1-Distill-Qwen-7B	0.25	0.24	0.15	0.18	0.27	0.15	0.28
Qwen3-4B	0.03	0.08	0.21	0.16	0.19	0.34	0.24

Table 1 reports $\mathbb{E}_{\boldsymbol{x}\sim P_\mathcal{D}}[\mathcal{U}_{\boldsymbol{x}}]$ results across systems under different aggregation strategies. *In 94% of cases (79 out of 84 groups),* $\mathbb{E}_{\boldsymbol{x}\sim P_\mathcal{D}}[\mathcal{U}_{\boldsymbol{x}}] < 1$*, indicating that the overscaling curse is prevalent in real-world systems.* Moreover, among these 79 cases, only one exceeds 0.5, while the remaining 78 are below 0.5. This means that, *for almost all affected systems, more than*

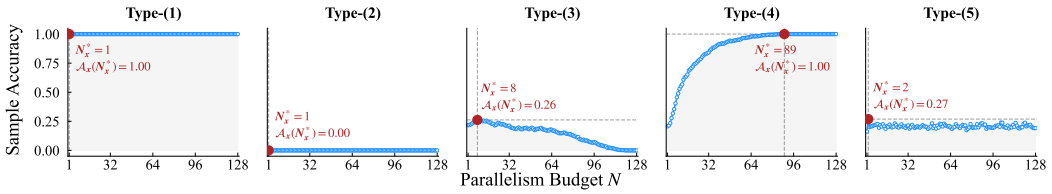


Figure 3: Illustrations of $\mathcal{A}_{\boldsymbol{x}}(N)$ from the five sample types based on the monotonic patterns. Examples here are from `Qwen3-4B` on the `AIME25` dataset. See Appendix C.1 for formal definitions.

half of the system-optimal budget is redundant for samples on average, showing that the overscaling curse is severe. Importantly, this phenomenon is consistent across aggregation strategies, suggesting that it is not an artifact of a specific aggregation but rather inherent to the system itself. These results consistently ***demonstrate the prevalence and severity of the overscaling curse in real-world systems***.

3.3 Better Understand the Overscaling Curse

After empirically verifying the overscaling curse, we trace backward from the outcome of this closed-loop process. Since $\mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}}}[\mathcal{U}_{\boldsymbol{x}}]$ depends on the distribution of $\mathcal{U}_{\boldsymbol{x}}$, and $\mathcal{U}_{\boldsymbol{x}}$ is determined by both $N^*_{\boldsymbol{x}}$ and the global $N^*_{\mathcal{D}}$, we further examine them in each system.

Figure 17 and 18 – 43 present the results. A highly imbalanced pattern consistently emerges: the distribution of $N^*_{\boldsymbol{x}}$ is heavily concentrated on the left side of the budget axis, indicating that most samples achieve their best accuracy with relatively small budgets. However, despite the sparse right region, $N^*_{\mathcal{D}}$ is often pulled toward that side by a few high-optimal-budget samples. As a result, most left-side samples are forced to operate under a much larger global budget. This leads to very low $\mathcal{U}_{\boldsymbol{x}}$ and consequently low $\mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}}}[\mathcal{U}_{\boldsymbol{x}}]$ over $\mathcal{D}$, thereby causing the severe overscaling curse.

This raises a further question: why does this phenomenon emerge and lead to such a severe overscaling curse? We further trace backward along the closed-loop process and revisit the sample-level budget-accuracy functions $\mathcal{A}_{\boldsymbol{x}}(N)$, as illustrated in Figure 1 and 6 – 16. Although sample behaviors are intertwined, most $\mathcal{A}_{\boldsymbol{x}}(N)$ exhibit clear monotonic patterns. We group these dominant patterns into four types: *(1)* $\mathcal{A}_{\boldsymbol{x}}(N) \equiv 1$*; (2)* $\mathcal{A}_{\boldsymbol{x}}(N) \equiv 0$*; (3)* $\mathcal{A}_{\boldsymbol{x}}(N)$ *approximately decreases toward* 0*; and (4)* $\mathcal{A}_{\boldsymbol{x}}(N)$ *approximately increases toward* 1. All remaining cases are labeled as *Type-5, covering non-monotonic functions without a clear trend*. Accordingly, $\mathcal{D}$ can be partitioned into five disjoint subsets $\{\mathcal{D}_i\}_{i=1}^5$, with $\mathcal{D}_i$ containing Type-i samples. For each $\mathcal{D}_i$, we denote its expected sample-optimal budget as $\mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}_i}}[N^*_{\boldsymbol{x}}]$. Formal definitions of these five types are in Appendix C.1.

Figure 3 presents one example illustration of $\mathcal{A}_{\boldsymbol{x}}(N)$ for the five types, with more illustrations shown in Figures 44 – 55. According to the monotonicity feature, samples in $\mathcal{D}_4$ naturally tend to have larger $N^*_{\boldsymbol{x}}$, especially compared to those from $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$, while $\mathcal{D}_5$ is less predictable due to its non-monotonicity. We empirically verify this by computing all $\mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}_i}}[N^*_{\boldsymbol{x}}]$ under each system (see Appendix C.2). The results confirm that $\mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}_4}}[N^*_{\boldsymbol{x}}]$ is substantially larger than that of the other types, including Type-5. This is the key inherent property for Type-4 samples in real-world systems.

Delving deeper, we analyze each subset $\mathcal{D}_i$ by its proportion p_i and total accuracy gain $\bar{\Delta}_i$ over the full budget domain. For each $\mathcal{D}_i$, let $p_i = |\mathcal{D}_i|/|\mathcal{D}|$ and $\bar{\Delta}_i = \mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}_i}}[\mathcal{A}_{\boldsymbol{x}}(N_{\max}) - \mathcal{A}_{\boldsymbol{x}}(1)]$. We find that *severe overscaling systems commonly satisfy* $p_4 > p_3$ *and* $\bar{\Delta}_4 > [-\bar{\Delta}_3]_+$*, where* $[z]_+ = \max(z, 0)$ (see Appendix C.2). Since dataset accuracy averages over all samples, the influence of each $\mathcal{D}_i$ for $\mathcal{A}_{\mathcal{D}}(N)$ is jointly determined by p_i and $\bar{\Delta}_i$. Thus, $\mathcal{D}_4$ dominates the positive improvement of $\mathcal{A}_{\mathcal{D}}(N)$, while $\mathcal{D}_3$ only weakly pulls it in the opposite direction; $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_5$ are negligible for providing zero or minimal gains. As a result, $N^*_{\mathcal{D}}$ is pulled toward the high-budget region near the large $\mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}_4}}[N^*_{\boldsymbol{x}}]$, while the remaining samples cannot pull it back. This explains why many $N^*_{\boldsymbol{x}}$ concentrate in the low-budget region , while $N^*_{\mathcal{D}}$ shifts rightward (Figure 2), causing serve overscaling. This intuition can be supported by the following theorem, with proof in Appendix C.3.

Theorem 1 *Let* $\mu_i = \mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}_i}}[N^*_{\boldsymbol{x}}]$*, where* $\mu_4 > \mu_i$ $(i \neq 4)$*. With monotonicity features, We have* $\bar{\Delta}_1 = \bar{\Delta}_2 = 0$ *and* $\bar{\Delta}_5 \approx 0$*. If* $p_4\bar{\Delta}_4 > p_3[-\bar{\Delta}_3]_+$*, then the budget utilization* $\mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}}}[\mathcal{U}_{\boldsymbol{x}}] < 1$.

So far, we have completed the backward tracing of the closed-loop process, and found that *the key source of the overscaling curse lies in Type-4 samples: their large accuracy gains pull* $N^*_{\mathcal{D}}$ *toward the high-budget region, while other samples cannot counteract this pull. Consequently, many low-*$N^*_{\boldsymbol{x}}$ *samples suffer reduced budget utilization, leading to low* $\mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}}}[\mathcal{U}_{\boldsymbol{x}}]$ *and thus the overscaling curse.*

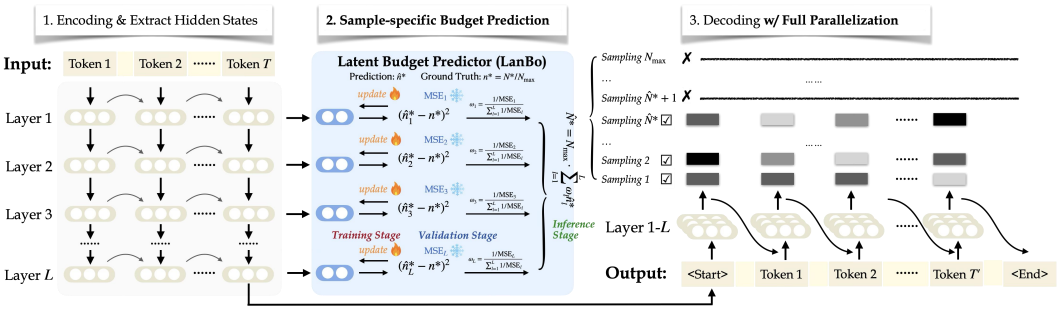


Figure 4: Overview of **our LATBP method (middle)** and the **new efficient parallel decoding paradigm PREADA (whole)**, formed by integrating LATBP into the full decoding pipeline.

4 Breaking the Overscaling Curse

Having demonstrated the prevalence and severity of the overscaling curse, we now seek methods to break it. Specifically, given a model π, for any dataset $\mathcal{D}$, we aim to *improve the sample efficiency (i.e., budget utilization) while maintaining overall system efficacy (i.e., dataset accuracy).*

4.1 Motivation

The overscaling curse is induced by enforcing a single global budget on all samples. Therefore, a natural solution is to relax this constraint and allocate sample-specific budgets. Since each sample has its own sample-optimal budget $N^*_{\boldsymbol{x}}$ (Eq.2), ideally, each sample should receive a budget close to $N^*_{\boldsymbol{x}}$. In this case, $\mathcal{U}_{\boldsymbol{x}}$ approaches 1, and the dataset accuracy is maintained or even improved, since $\mathbb{E}[\mathcal{A}_{\boldsymbol{x}}(N^*_{\boldsymbol{x}})] \geq \mathbb{E}[\mathcal{A}_{\boldsymbol{x}}(N_{\mathcal{D}})]$. Thus, breaking the curse reduces to predicting $N^*_{\boldsymbol{x}}$ for each sample.

This raises the immediate question: what features should we use to predict $N^*_{\boldsymbol{x}}$? Intuitively, for a given sample $\boldsymbol{x}$, a very strong or very weak model may only need a small budget, as it tends to be consistently right or wrong regardless of sampling. A moderately capable model, however, may require more sampling to reach a reliable decision. Thus, $N^*_{\boldsymbol{x}}$ depends on both $\boldsymbol{x}$ and how the model π perceives it. This motivates using *model-aware features*: The hidden states of π when encoding $\boldsymbol{x}$ naturally capture this information, making them suitable for predicting $N^*_{\boldsymbol{x}}$. In contrast, methods that rely only on the raw sample, such as few-shot learning with external LLMs, may miss these signals.

4.2 Method: Latent Budget Predictor (LATBP, Figure 4)

With the above considerations, for a given model π, we train a lightweight budget predictor that learns to predict $N^*_{\boldsymbol{x}}$ from π's hidden state of $\boldsymbol{x}$ through real-world samples. Once trained, this predictor should generalize to unseen samples, producing budget predictions for them during inference.

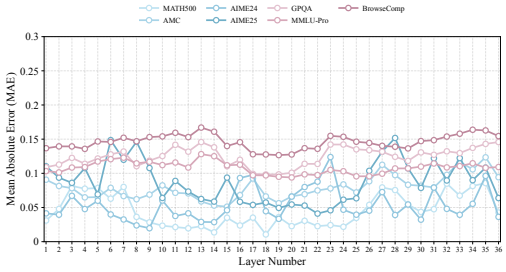
Assume π has L decoder layers. For an input $\boldsymbol{x} = (x_1, \ldots, x_T)$ of length T, the token embeddings $\boldsymbol{H}^0 \in \mathbb{R}^{T\times d}$ are processed layer by layer: $\boldsymbol{H}^l = F^l(\boldsymbol{H}^{l-1})$, where d is the hidden dimension and F^l is the l-th layer function. Due to causal attention, the last-token representation $\boldsymbol{h}^l_T = \boldsymbol{H}^l[:, T]$ can serve as an information aggregation of the entire input at layer l [2, 47, 10].

Training: Layer-wise Learning. We first consider the training process. Since we do not know in advance which layer's hidden state is most suitable as a learnable feature, we use all $\boldsymbol{h}^l_T$ from each layer and train L layer-wise predictors $\{\phi^\pi_{\theta_l} : \mathbb{R}^d \to \mathbb{R}\}_{l=1}^L$. Each predictor is defined as $\phi^\pi_{\theta_l}(\boldsymbol{h}^l_T(\boldsymbol{x})) = N^*_{\boldsymbol{x}}/N_{\max}$, where the target is normalized as $n^*_{\boldsymbol{x}} = N^*_{\boldsymbol{x}}/N_{\max} \in [0, 1]$ to stabilize training. Each $\phi^\pi_{\theta_l}$ is trained on the training data $\mathcal{D}_{\text{train}}$ to minimize the squared loss:

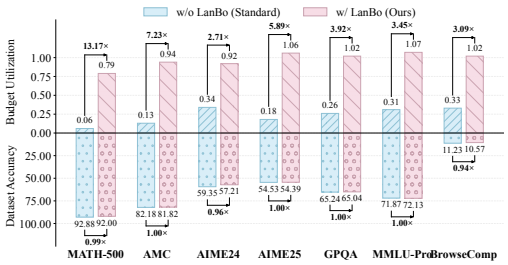
$$\mathcal{L}^\pi_l = \mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}_{\text{train}}}}\left[\left(\phi^\pi_{\theta_l}(\boldsymbol{h}^l_T(\boldsymbol{x})) - n^*_{\boldsymbol{x}}\right)^2\right], \quad \text{for } 1 \leq l \leq L, \tag{4}$$

Inference: Layer-wise Combination. After training, given an unseen $\boldsymbol{x}$, we can obtain L outputs from the L learned predictors. The remaining challenge is how to combine them into a final prediction.

Let $\hat{n}^*_{\boldsymbol{x},l} = \phi^\pi_{\theta_l}(\boldsymbol{h}^l_T(\boldsymbol{x}))$ denote the l-th predictor output. We obtain the final prediction by linearly combining all layer-wise outputs with weights ω_l at layer l:



(a) MAE results of the L layer-wise predictors on the validation set, with each averaged *over 8 runs*.



(b) Comparison of expected budget utilization and dataset accuracy before and after applying LATBP.

Figure 5: **Empirical Validation of LATBP for Breaking the Overscaling Curse**. Here we use `Qwen3-4B` with *voting* aggregation, with other results presented in Figures 56 – 62 in Appendix E.

$$\hat{N}^*_{\boldsymbol{x}} = N_{\max} \cdot \boldsymbol{\omega}^\top \hat{\boldsymbol{n}}^*_{\boldsymbol{x}}, \qquad \text{s.t. } \mathbf{1}^\top \boldsymbol{\omega} = 1, \tag{5}$$

where $\hat{\boldsymbol{n}}^*_{\boldsymbol{x}} = (\hat{n}^*_{\boldsymbol{x},1}, \ldots, \hat{n}^*_{\boldsymbol{x},L})^\top$ and $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_L)^\top$. Although the ideal weights are not directly available, our inference objective is clear: *minimize the expected error of the final prediction on unseen samples*. Thus, the optimal weights can be solved by the following optimization problem

$$\boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\omega}} \ \mathbb{E}_{\boldsymbol{x} \sim P_{\mathcal{D}_{\text{unseen}}}} \left[\left(\boldsymbol{\omega}^\top \hat{\boldsymbol{n}}^*_{\boldsymbol{x}} - n^*_{\boldsymbol{x}} \right)^2 \right] \qquad \text{s.t. } \mathbf{1}^\top \boldsymbol{\omega} = 1. \tag{6}$$

The solution to this optimization problem is $\boldsymbol{\omega}^* = \frac{\Sigma^{-1}\mathbf{1}}{\mathbf{1}^\top \Sigma^{-1} \mathbf{1}}$, where $\Sigma = \mathbb{E}_{\boldsymbol{x} \sim P_{\mathcal{D}}}[(\hat{\boldsymbol{n}}^*_{\boldsymbol{x}} - n^*_{\boldsymbol{x}}\mathbf{1})(\hat{\boldsymbol{n}}^*_{\boldsymbol{x}} - n^*_{\boldsymbol{x}}\mathbf{1})^\top]$ is the error covariance matrix across layers. The derivation is provided in Appendix D.2. In principle, computing $\boldsymbol{\omega}^*$ requires the full covariance matrix Σ, but estimating and inverting it can be unstable in practice. We thus adopt a diagonal surrogate $\Sigma \approx \operatorname{diag}(m_1, \ldots, m_L)$, where $m_l = \mathbb{E}_{\boldsymbol{x} \sim P_{\mathcal{D}_{\text{unseen}}}}[(\hat{n}^*_{\boldsymbol{x},l} - n^*_{\boldsymbol{x}})^2]$ is the second moment of the prediction error of the l-th predictor. The approximation error is analyzed in Appendix D.2, which is ignorable in practice.

Under this approximation, the optimal weight vector reduces to inverse-error weighting: $\boldsymbol{\omega} \propto \boldsymbol{m}^{-1}$, where $\boldsymbol{m} = (m_1, \ldots, m_L)^\top$. Since the true distribution is unknown, we introduce a ***Validation Set*** $\mathcal{D}_{\textbf{\textit{val}}}$ and estimate m_l using the empirical MSE $\hat{m}_l$ of l-th predictor. The final weights are then:

$$\boldsymbol{\omega} = \hat{\boldsymbol{m}}^{-1} / \|\hat{\boldsymbol{m}}^{-1}\|_1, \quad \text{where } \hat{\boldsymbol{m}}^{-1} = (1/\hat{m}_1, \ldots, 1/\hat{m}_L)^\top. \tag{7}$$

Applying this to Eq.5 yields the final prediction $\hat{N}^*_{\boldsymbol{x}}$. This scheme actually assigns higher weights to predictors with lower squared error, aligning with the intuition that better should contribute more.

4.3 General Implementation

Having described the general training and inference procedures, we now specify the implementation. For the architecture, we follow the principle of minimal algebraic structure to employ a lightweight network: a single-hidden-layer MLP with a ReLU activation, followed by a logistic regression output layer to adapt the regression objective. We have also tested more complex architectures like deeper MLPs and shallow Transformers, but found no clear performance gains. We therefore adopt the simplest architecture that provides effective learning with minimal cost and no overfitting. Let d be the hidden dimension of the language model, we set the hidden size of MLP as $\lfloor rd \rfloor$ with $r = 1/8$.

For training data[5], we consider two main dimensions: *difficulty diversity* and *domain diversity*. The former enables the predictors to learn how to allocate budgets to problems of varying difficulty, which is the primary learning objective. The latter enhances their adaptability across multiple domains, facilitating generalization to unseen scenarios. The validation data is drawn from the same source as the training data. Detailed data specifications and training procedures are provided in Appendix D.1.

4.4 Empirical Validation

Evaluation Metric. We first check whether the L predictors are well learned by measuring their Mean Absolute Error (MAE) on the test sets. Next, we formally evaluate whether LATBP breaks the

[5]Our training data are from open-source corpus and contain no samples in our evaluation sets. We also do not filter by domain to match our evaluation domains, ensuring reliable subsequent conclusions about domain adaptability.

overscaling curse. Specifically, We replace the global budget with the sample-specific prediction of the actual budget $N_{\boldsymbol{x}}$ in the budget utilization $\mathcal{U}_{\boldsymbol{x}}$ (Eq.3), and compare the expected budget utilization $\mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}}}[\mathcal{U}_{\boldsymbol{x}}]$ before and after the replacement, along with the corresponding dataset accuracy.

Main Results (Figure 5). 5(a) shows the learning results. MAEs on the four mathematical datasets are lowest (mostly below 0.1). On the other three domains, although MAEs are slightly higher than those on mathematical ones, they remain around 0.15, still indicating good effectiveness. Across layers, intermediate layers are more stable than early or late ones, which exhibit larger fluctuations, suggesting that intermediate representations offer more robust features, consistent with [42]. ***Overall,* LATBP *demonstrates strong learning effectiveness and good domain adaptability.***

5(b) presents the inference results. Clearly, after applying our LATBP, the budget utilization increases significantly, hovering around 1, demonstrating that sample efficiency has been greatly improved. In terms of performance, there is almost no significant degradation; in fact, performance even improves on certain datasets, satisfying the condition of maintaining system efficacy. Meanwhile, this phenomenon holds across all datasets, indicating that our method is effective across various domains. ***Overall,* LATBP *effectively breaks the overscaling curse and adapts well to diverse domains.***

5 Practical Application: Real-world Efficient Parallel Decoding

In the previous section, we revealed the overscaling curse and introduced LATBP as a solution. While LATBP improves efficiency in terms of budget utilization, it must ultimately be deployed in real-world decoding pipelines to demonstrate practical usability. In practice, efficiency depends not only on simple budget size but also on hardware-aware costs such as inference latency and memory overhead [44, 50]. Therefore, in this section, we further integrate LATBP into the full parallel decoding pipeline and evaluate whether it brings practical efficiency gains beyond budget reduction.

5.1 New Paradigm: Pre-decoding Budget Adaptation (PREADA)

When integrating LATBP into the full decoding pipeline, the resulting pipeline is illustrated in Figure 4, with the following three steps: (1) The model π encodes input $\boldsymbol{x}$ and extracts hidden states $\boldsymbol{h}_T^l(\boldsymbol{x})$ from all layers; (2) The well-trained L predictors produce a budget prediction $\hat{N}_{\boldsymbol{x}}^*$; (3) $\hat{N}_{\boldsymbol{x}}^*$ sequences are sampled with full hardware parallelization and aggregated to the final answer.

Table 2: Extra blocking comparisons of two efficient parallel decoding paradigms.

	Extra Training	Hardware Blocking	
		Before Decoding	During Decoding
INADA	✗	✗	✓
PREADA (Ours)	✓	✓	✗

Highlighted Comparison. The key feature of our pipeline is that budget allocation is performed *before decoding*. Existing efficient parallel decoding methods also pursue sample-specific budgets, but typically adapt them *during decoding*. They often split the original large parallel batch into smaller mini-batches, sometimes of size 1, and decide after each mini-batch whether to stop or continue using internal confidence signals [26, 12] or external model signals [48]. Based on this distinction, LATBP naturally inspires a new efficient parallel decoding paradigm, **Pre-decoding Budget Adaptation (PREADA)**. Correspondingly, we refer to existing methods as the *Decoding-time Budget Adaptation (*INADA*)* paradigm. ***Both paradigms aim at adaptive budgeting, but differ in when the budget is determined.***

Despite these differences, both paradigms introduce extra blocking costs to the decoding pipeline when reducing budgets. As in Table 2, INADA blocks full hardware parallelization during decoding, while PREADA incurs cost for predictor training and inference before decoding. Therefore, comparing methods under the two paradigms with hardware-aware efficiency can truly reveal the practical value of our LATBP, and demonstrate the promise of the new PREADA paradigm it represents.

5.2 Empirical Validation

Baseline. Considering the extra cost of verifiers, we use majority voting as the aggregation strategy [49]. It is lightweight and is widely used in parallel decoding. The standard parallel decoding baseline (*STD.*) therefore also uses majority voting, with its global budget set to $N_{\mathcal{D}}^*$ to maximize dataset accuracy. For the INADA paradigm, we evaluate four representative methods: ASC [1], ESC [26], DSC [48], and the online version of DeepConf [12]. Details are provided in Appendix F. The predictors in our LATBP is implemented consistently with Section 4.3.

Table 3: **Empirical Validation of Our LatBP for Efficient Parallel Decoding.** Under each system, we report the real-world efficiency (memory $\delta_{\mathcal{M}}$ and latency $\delta_{\mathcal{T}}$, both the ratio relative to *STD.*) and accuracy (*Acc.*), where *STD.* denotes the standard parallel decoding. **Bold** and underline denote the best and second best, respectively. Blue means higher accuracy than *STD.*; red means lower efficiency. Each result is averaged ***over 32 runs***, with standard deviations in Appendix G.2.

Paradigm	Method	MATH500			AMC			AIME24			AIME25			GPQA			MMLU-Pro			BrowseComp		
		$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑	$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑	$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑	$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑	$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑	$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑	$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑
`Qwen2.5-7B`																						
—	*STD.*	*1.00*	*1.00*	*78.17*	*1.00*	*1.00*	*47.18*	*1.00*	*1.00*	*20.10*	*1.00*	*1.00*	*13.58*	*1.00*	*1.00*	*39.49*	*1.00*	*1.00*	*60.18*	*1.00*	*1.00*	*3.21*
InAda	AC	0.24	1.49	77.26	0.28	2.32	46.67	0.74	3.46	19.65	0.66	2.83	11.94	0.38	2.70	38.79	0.32	2.03	59.28	0.40	1.76	2.58
	ESC	0.52	2.77	77.58	0.34	1.90	46.77	0.32	0.96	19.72	0.33	1.45	12.02	0.36	2.27	39.05	0.28	1.78	59.38	0.35	1.65	2.71
	DSC	0.44	1.43	77.33	0.31	2.16	46.85	**0.27**	1.83	18.52	**0.28**	0.97	9.89	0.31	2.18	38.12	0.40	3.74	59.59	0.41	3.12	1.92
	DeepConf	0.45	3.45	**77.82**	0.38	2.67	46.67	1.13	3.16	18.08	0.78	1.88	12.25	0.53	3.41	**39.19**	0.61	3.21	59.63	0.65	2.18	3.00
PreAda	LatBP (Ours)	**0.18**	**0.48**	77.49	**0.23**	**0.74**	**47.08**	0.49	**0.52**	**20.91**	0.35	**0.68**	**13.84**	**0.27**	**0.51**	38.32	**0.23**	**0.39**	**59.85**	**0.30**	**0.49**	**3.17**
`Llama3.1-8B`																						
—	*STD.*	*1.00*	*1.00*	*57.43*	*1.00*	*1.00*	*26.87*	*1.00*	*1.00*	*10.68*	*1.00*	*1.00*	*0.86*	*1.00*	*1.00*	*36.66*	*1.00*	*1.00*	*52.54*	*1.00*	*1.00*	*0.56*
InAda	AC	**0.18**	1.71	55.90	0.21	2.27	25.73	0.34	3.56	9.85	0.41	3.50	0.78	0.37	2.99	36.24	0.31	3.17	52.58	0.34	2.12	0.51
	ESC	0.34	1.40	**56.65**	0.53	2.03	25.56	0.33	1.64	11.28	0.21	2.44	0.69	0.32	2.38	36.45	0.35	2.16	**52.63**	0.50	1.88	0.50
	DSC	0.28	1.75	55.38	0.33	2.18	25.56	0.27	1.63	7.98	**0.14**	0.98	0.35	0.26	2.08	35.92	0.26	2.83	52.00	0.35	2.44	0.43
	DeepConf	0.44	2.37	56.12	0.48	2.61	25.82	0.67	4.02	7.43	0.72	5.22	0.47	0.52	4.17	34.49	0.63	4.16	49.86	0.66	3.19	0.44
PreAda	LatBP (Ours)	0.23	**0.41**	56.62	**0.16**	**0.37**	**26.69**	**0.22**	**0.43**	**11.39**	0.17	**0.52**	**0.80**	**0.25**	**0.37**	**36.48**	**0.22**	**0.42**	52.23	**0.26**	**0.45**	**0.62**
`Deepseek-R1-Distill-Qwen-7B`																						
—	*STD.*	*1.00*	*1.00*	*94.24*	*1.00*	*1.00*	*86.78*	*1.00*	*1.00*	*79.11*	*1.00*	*1.00*	*68.67*	*1.00*	*1.00*	*55.29*	*1.00*	*1.00*	*76.02*	*1.00*	*1.00*	*17.45*
InAda	AC	0.28	1.89	93.70	0.32	2.77	85.82	0.44	2.50	78.64	0.37	2.75	67.79	0.54	2.62	54.67	0.47	2.32	75.72	**0.28**	1.76	16.83
	ESC	**0.15**	0.84	94.02	0.27	1.72	**86.03**	0.53	2.19	78.39	0.41	2.19	68.12	0.43	1.67	54.42	0.38	1.92	75.37	0.45	1.92	14.23
	DSC	0.45	2.13	91.04	0.39	2.89	81.28	0.56	3.58	73.75	0.32	2.00	65.30	0.60	4.03	52.99	0.35	1.79	71.14	0.46	2.11	15.23
	DeepConf	0.53	3.86	93.45	0.58	3.56	85.52	0.83	4.98	**79.58**	0.75	3.57	**68.90**	0.65	3.80	53.13	0.59	3.88	73.40	0.66	3.12	16.09
PreAda	LatBP (Ours)	0.19	**0.37**	**94.32**	**0.22**	**0.48**	85.97	**0.34**	**0.71**	79.30	**0.26**	**0.50**	68.28	**0.25**	**0.45**	**55.53**	**0.23**	**0.49**	**75.84**	0.31	**0.52**	**16.96**
`Qwen3-4B`																						
—	*STD.*	*1.00*	*1.00*	*92.88*	*1.00*	*1.00*	*82.18*	*1.00*	*1.00*	*59.35*	*1.00*	*1.00*	*54.53*	*1.00*	*1.00*	*65.24*	*1.00*	*1.00*	*71.87*	*1.00*	*1.00*	*11.23*
InAda	AC	0.18	1.56	**92.17**	0.14	2.67	81.73	0.84	3.12	57.81	0.26	2.41	53.81	**0.21**	2.11	65.03	0.30	2.36	71.72	**0.26**	1.90	10.51
	ESC	**0.12**	0.88	92.11	0.23	1.84	81.77	**0.55**	1.93	57.34	0.28	2.25	53.86	0.32	1.97	65.12	0.34	2.23	71.50	0.34	2.34	9.80
	DSC	0.16	1.35	91.17	0.28	1.71	81.32	0.72	2.21	57.18	0.22	2.78	50.23	0.36	3.92	64.21	0.39	2.97	70.38	0.45	2.67	10.00
	DeepConf	0.29	2.77	91.87	0.32	3.27	81.65	1.19	4.63	**59.30**	0.24	3.76	52.97	0.49	3.20	64.87	0.58	4.99	72.15	0.62	4.23	10.11
PreAda	LatBP (Ours)	0.20	**0.45**	92.07	**0.10**	**0.52**	**81.88**	0.57	**0.79**	59.29	**0.17**	**0.56**	**54.55**	0.27	**0.46**	**65.17**	**0.22**	**0.40**	**72.20**	0.28	**0.48**	**10.63**

Evaluation Metric. When evaluating in a system $(\pi, \mathcal{D})$, we still first evaluate the dataset accuracy. For efficiency, we use two hardware-aware metrics: *memory overhead* and *inference latency* [50]:

- For memory, we measure the peak usage per sample $\boldsymbol{x}$, which reflects the required hardware capacity. This usage is dominated by the KV cache and model weights, *plus any extra overhead from extra models*. We compute the expected memory peak usage $\mathcal{M}_{\text{met}}$ per sample over $\mathcal{D}$.
- For latency, we measure wall-clock time per sample $\boldsymbol{x}$ from encoding to answer aggregation, including all intermediate steps. We compute the total latency $\mathcal{T}_{\text{met}}$ over $\mathcal{D}$. *For our* LatBP*, we include training time in each system evaluation to ensure fair comparison.*

We compare each method against *STD.* using relative ratios $\delta_{\mathcal{T}} = \mathcal{T}_{\text{met}}/\mathcal{T}_{\text{std.}}$ and $\delta_{\mathcal{M}} = \mathcal{M}_{\text{met}}/\mathcal{M}_{\text{std.}}$. Smaller δ indicates better optimization, and δ above 1 implies no real improvement over *STD.*

Main Results (Table 3). In terms of efficiency, LatBP achieves over 50% memory savings in most settings, ranks among the top two in over 90% of cases, and remains comparable to other methods. More importantly, its inference latency is significantly lower than that of the others. The key reason is that *decoding-time parallelization blocking is more costly than expected*. With strong parallel acceleration such as `vLLM`, a single parallel request with $N = 128$ incurs only about 2–3× the latency of $N = 1$, and this gap shrinks further for longer reasoning. Therefore, *for* InAda *methods, serializing parallel requests can hurt latency more than budget reduction helps*, sometimes even making them slower than *STD.* In contrast, *keeping decoding fully parallelized is key to* LatBP*'s low latency*. Meanwhile, the pre-decoding predictor is lightweight, adding only milliseconds of inference overhead. Even including training time, the total time is only tens of seconds, remaining modest compared with the total decoding time. See Appendix G.3 for detailed latency statistics. Overall, **LatBP *enables truly efficient parallel decoding in real-world settings.***

In terms of accuracy, LatBP maintains stable performance compared with *STD.*, with overall fluctuations within 2%, and even outperforms *STD.* in about 30% of cases. Compared with InAda methods, LatBP consistently achieves the best or second-best accuracy. This stability arises from directly learning optimal budget features, thereby avoiding unstable heuristic strategies used by InAda methods. For example, DSC relies on an extra model to rank problem difficulty, while DeepConf uses model confidence; both perform well on `R1` but are less effective on `Llama3.1`. Thus, **LatBP *maintains stable accuracy, satisfying the prerequisite for efficient parallel decoding.***

Furthermore, ***we emphasize that* LatBP *exhibits strong domain adaptability.*** Even though the predictor for each model is trained only once, it performs consistently well across diverse domains. This further demonstrates the completeness of our training and enhances its practical value.

6 Extended Analysis

Sensitivity to Sampling Hyperparameters. In the main experiments, we use a fixed set of sampling hyperparameters (Section 2.1). We now evaluate the sensitivity of LATBP to hyperparameter variations under two settings: *(1) In-Domain (ID) Adaptability*, where training and decoding use the same changed configuration; and *(2) Out-of-Domain (OOD) Generalization*, where the predictor is trained with one configuration but another is used for decoding. Results are in Appendix H.1.

Ablation Study of LATBP. We perform three ablations: *(1) MLP hidden size*, *(2) training data size*, and *(3) inference weighting strategy*. Although the third is derived theoretically, we compare it with common alternatives to validate its practical soundness. Results are shown in Appendix H.2.

Design Principles of LATBP. LATBP also involves two implicit design principles: *(1) training objective*, where we use continuous regression rather than discrete classification; and *(2) layer-wise pattern*, where we train layer-wise L predictors and combine them at inference, rather than concatenating hidden states from all L layers into a single predictor. We discuss the rationale behind these choices and empirically verify their rationality in Appendix H.3.

7 Conclusion and Future Work

This paper reveals the overscaling curse in parallel thinking. A global budget selected to maximize system efficacy can substantially reduce sample budget utilization, undermining sample efficiency. We empirically show that this contradiction is both prevalent and severe in real-world systems. To break the overscaling curse, we propose LATBP, which probes the model's latent representations of each sample to predict its sample-optimal budget. LATBP significantly improves budget utilization while maintaining dataset accuracy. Finally, we integrate LATBP into the full decoding pipeline, inspiring a new efficient decoding paradigm, PREADA, which substantially reduces latency and memory usage, demonstrating its practical usability. More **related works** are provided in Appendix I.

Looking ahead from this work, our proposed PREADA is a promising paradigm for efficient parallel decoding, as it avoids decoding-time blocking and preserves full hardware parallelization. In this work, LATBP serves as a first instantiation of this paradigm. Although effective, it still leaves substantial room for further improvement. For example, future work can explore training-free budget prediction methods, thereby further reducing overhead and improving efficiency.

Limitations

First, we do not consider open-ended tasks, since the lack of reliable ground truth makes it difficult to define the sample-optimal budget. Indeed, evaluating open-ended outputs remains an open challenge. Once suitable evaluation protocols become available, LATBP can be naturally extended to such settings. Second, LATBP requires access to internal model states, and therefore currently applies mainly to open-source models. However, with the rapid growth of the open-source LLM ecosystem [60, 17], we believe that this limitation does not diminish our contribution.

Broader Impacts

This work aims to improve the efficiency of parallel thinking by reducing redundant budget. Its positive impact is that more efficient budget allocation can reduce inference latency, memory usage, and computational cost, making advanced reasoning systems more accessible under limited hardware resources. We do not identify negative societal impacts specific to this work. Our method does not introduce new model capabilities; it only reallocates inference budgets for existing parallel decoding pipelines. Potential risks mainly come from the underlying LLMs and their downstream applications, rather than from the budget prediction method itself. Therefore, responsible deployment should follow the safety and evaluation practices required for the base models and target applications.

Acknowledgements

This paper is funded by the Frontier Technologies R&D Program of Jiangsu.

References

[1] Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. Let's Sample Step by Step: Adaptive-Consistency for Efficient Reasoning and Coding with LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12375–12396, 2023.

[2] Amos Azaria and Tom Mitchell. The Internal State of an LLM Knows When It's Lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 967–976, 2023.

[3] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large Language Monkeys: Scaling Inference Compute with Repeated Sampling. *arXiv preprint arXiv:2407.21787*, 2024.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901, 2020.

[5] Lingjiao Chen, Jared Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. Are More LLM Calls All You Need? Towards the Scaling Properties of Compound AI Systems. In *Advances in Neural Information Processing Systems*, volume 37, pages 45767–45790, 2024.

[6] Mouxiang Chen, Binyuan Hui, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Jianling Sun, Junyang Lin, and Zhongxin Liu. Parallel Scaling Law for Language Models. In *Advances in Neural Information Processing Systems*, volume 38, pages 118958–118998, 2025.

[7] Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do NOT Think That Much for 2+3=? On the Overthinking of Long Reasoning Models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pages 9487–9499, 2025.

[8] Yifu Ding, Wentao Jiang, Shunyu Liu, Yongcheng Jing, Jinyang Guo, Yingjie Wang, Jing Zhang, Zengmao Wang, Ziwei Liu, Bo Du, Xianglong Liu, and Dacheng Tao. Dynamic Parallel Tree Search for Efficient LLM Reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11233–11252, 2025.

[9] Harry Dong, David Brandfonbrener, Eryk Helenowski, Yun He, Mrinal Kumar, Han Fang, Yuejie Chi, and Karthik Abinav Sankararaman. Generalized Parallel Scaling with Interdependent Generations. In *International Conference on Learning Representations*, pages 43631–43646, 2026.

[10] Zhichen Dong, Zhanhui Zhou, Zhixuan Liu, Chao Yang, and Chaochao Lu. Emergent Response Planning in LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pages 14301–14320, 2025.

[11] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical Neural Story Generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, 2018.

[12] Yichao Fu, Xuewei Wang, Hao Zhang, Yuandong Tian, and Jiawei Zhao. Deep Think with Confidence. In *International Conference on Learning Representations*, pages 94355–94377, 2026.

[13] Kanishk Gandhi, Ayush K. Chakravarthy, Anikait Singh, Nathan Lile, and Noah Goodman. Cognitive Behaviors that Enable Self-Improving Reasoners, or, Four Habits of Highly Effective STaRs. In *Second Conference on Language Modeling*, 2025.

[14] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, et al. Gemini: A Family of Highly Capable Multimodal Models. *arXiv preprint arXiv:2312.11805*, 2023.

[15] 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.

[16] Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rStar-Math: Small LLMs Can Master Math Reasoning with Self-Evolved Deep Thinking. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pages 20640–20661, 2025.

[17] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

[18] Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. DeepMath-103K: A Large-Scale, Challenging, Decontaminated, and Verifiable Mathematical Dataset for Advancing Reasoning. In *International Conference on Learning Representations*, pages 138306–138322, 2026.

[19] 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 *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

[20] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*, 2015.

[21] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The Curious Case of Neural Text Degeneration. In *International Conference on Learning Representations*, 2020.

[22] Colin Hong, Xu Guo, Anand Chaanan Singh, Esha Choukse, and Dmitrii Ustiugov. Slim-SC: Thought Pruning for Efficient Scaling with Self-Consistency. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 34500–34517, 2025.

[23] Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. LLM-Blender: Ensembling Large Language Models with Pairwise Ranking and Generative Fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14165–14178, 2023.

[24] Zhewei Kang, Xuandong Zhao, and Dawn Song. Scalable Best-of-N Selection for Large Language Models via Self-Certainty. In *Advances in Neural Information Processing Systems*, volume 38, pages 19720–19745, 2025.

[25] Baixuan Li, Dingchu Zhang, Jialong Wu, Wenbiao Yin, Zhengwei Tao, Yida Zhao, Liwen Zhang, Haiyang Shen, Runnan Fang, Pengjun Xie, Jingren Zhou, and Yong Jiang. ParallelMuse: Agentic Parallel Thinking for Deep Information Seeking. *arXiv preprint arXiv:2510.24698*, 2025.

[26] 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. In *International Conference on Learning Representations*, pages 14751–14768, 2024.

[27] Yizhi Li, Qingshui Gu, Zhoufutu Wen, Ziniu Li, Ruibin Yuan, Tianshun Xing, Shuyue Guo, Tuney Zheng, Xin Zhou, Xingwei Qu, Wangchunshu Zhou, Zheng Zhang, Wei Shen, Wei Xue, Qian Liu, Chenghua Lin, Jian Yang, Ge Zhang, and Wenhao Huang. TreePO: Enhancing Policy Efficacy and Inference Efficiency with Tree Modeling. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306, pages 66963–66976, 2026.

[28] Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging Divergent Thinking in Large Language Models through Multi-Agent Debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17889–17904, 2024.

[29] 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 *International Conference on Learning Representations*, pages 39578–39601, 2024.

[30] Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, and Yang Liu. Skywork-Reward-V2: Scaling Preference Data Curation via Human-AI Synergy. In *International Conference on Learning Representations*, pages 133805–133838, 2026.

[31] Xueguang Ma, Qian Liu, Dongfu Jiang, Ge Zhang, Zejun MA, and Wenhu Chen. General-Reasoner: Advancing LLM Reasoning Across All Domains. In *Advances in Neural Information Processing Systems*, volume 38, pages 56596–56618, 2025.

[32] Mathematical Association of America. American Mathematics Competitions (AMC). https://maa.org/student-programs/amc/.

[33] Mathematical Association of America. American Invitational Mathematics Examination (AIME). https://maa.org/maa-invitational-competitions/, 2024.

[34] Mathematical Association of America. American Invitational Mathematics Examination (AIME). https://maa.org/maa-invitational-competitions/, 2025.

[35] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Fei-Fei Li, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20275–20321, 2025.

[36] Xuefei Ning, Zinan Lin, Zixuan Zhou, Zifu Wang, Huazhong Yang, and Yu Wang. Skeleton-of-Thought: Prompting LLMs for Efficient Parallel Generation. In *International Conference on Learning Representations*, pages 917–967, 2024.

[37] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*, 2023.

[38] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A Graduate-Level Google-Proof Q&A Benchmark. In *First Conference on Language Modeling*, 2024.

[39] Gleb Rodionov, Roman Garipov, Alina Shutova, George Yakushev, Erik Schultheis, Vage Egiazarian, Anton Sinitsin, Denis Kuznedelev, and Dan Alistarh. Hogwild! Inference: Parallel LLM Generation via Concurrent Attention. In *Advances in Neural Information Processing Systems*, volume 38, pages 46592–46633, 2025.

[40] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*, 2024.

[41] Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. OpenAI GPT-5 System Card. *arXiv preprint arXiv:2601.03267*, 2025.

[42] Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Nikul Patel, Jalal Naghiyev, Yann Lecun, and Ravid Shwartz-Ziv. Layer by Layer: Uncovering Hidden Representations in Language Models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pages 55854–55875, 2025.

[43] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM Test-Time Compute Optimally Can be More Effective than Scaling Parameters for Reasoning. In *International Conference on Learning Representations*, pages 10131–10165, 2025.

[44] Hanshi Sun, Momin Haider, Ruiqi Zhang, Huitao Yang, Jiahao Qiu, Ming Yin, Mengdi Wang, Peter Bartlett, and Andrea Zanette. Fast Best-of-N Decoding via Speculative Rejection. In *Advances in Neural Information Processing Systems*, volume 37, pages 32630–32652, 2024.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[46] Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. Interpretable Preferences via Multi-Objective Reward Modeling and Mixture-of-Experts. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10582–10592, 2024.

[47] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving Text Embeddings with Large Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11897–11916, 2024.

[48] 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 *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 6919–6932, 2025.

[49] 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. In *International Conference on Learning Representations*, 2023.

[50] Yiming Wang, Pei Zhang, Siyuan Huang, Baosong Yang, Zhuosheng Zhang, Fei Huang, and Rui Wang. Sampling-Efficient Test-Time Scaling: Self-Estimating the Best-of-N Sampling in Early Decoding. In *Advances in Neural Information Processing Systems*, volume 38, pages 162137–162174, 2025.

[51] Yiming Wang, Pei Zhang, Jialong Tang, Hao-Ran Wei, Baosong Yang, Rui Wang, Chenshu Sun, Feitong Sun, Jiran Zhang, Junxuan Wu, Qiqian Cang, Yichang Zhang, Junyang Lin, Fei Huang, and Jingren Zhou. PolyMath: Evaluating Mathematical Reasoning in Multilingual Contexts. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

[52] Yiming Wang, Pei Zhang, Baosong Yang, Derek Wong, and Rui Wang. Latent Space Chain-of-Embedding Enables Output-free LLM Self-Evaluation. In *International Conference on Learning Representations*, pages 70938–70970, 2025.

[53] Yiming Wang, Pei Zhang, Baosong Yang, Derek Wong, Zhuosheng Zhang, and Rui Wang. Embedding Trajectory for Out-of-Distribution Detection in Mathematical Reasoning. In *Advances in Neural Information Processing Systems*, volume 37, pages 42965–42999, 2024.

[54] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark. In *Advances in Neural Information Processing Systems*, volume 37, pages 95266–95290, 2024.

[55] Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. BrowseComp: A Simple Yet Challenging Benchmark for Browsing Agents. *arXiv preprint arXiv:2504.12516*, 2025.

[56] Hao Wen, Yifan Su, Feifei Zhang, Yunxin Liu, Yunhao Liu, Ya-Qin Zhang, and Yuanchun Li. ParaThinker: Native Parallel Thinking as a New Paradigm to Scale LLM Test-time Compute. *arXiv preprint arXiv:2509.04475*, 2025.

[57] Sam Wiseman and Alexander M. Rush. Sequence-to-Sequence Learning as Beam-Search Optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306, 2016.

[58] Tong Wu, Yang Liu, Jun Bai, Zixia Jia, Shuyi Zhang, Ziyong Lin, Yanting Wang, Song-Chun Zhu, and Zilong Zheng. Native Parallel Reasoner: Reasoning in Parallelism via Self-Distilled Reinforcement Learning. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306, pages 136411–136426, 2026.

[59] Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. Can LLMs Express Their Uncertainty? An Empirical Evaluation of Confidence Elicitation in LLMs. In *International Conference on Learning Representations*, pages 23650–23678, 2024.

[60] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025.

[61] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*, 2024.

[62] Xinyu Yang, Yuwei An, Hongyi Liu, Tianqi Chen, and Beidi Chen. Multiverse: Your Language Models Secretly Decide How to Parallelize and Merge Generation. In *Advances in Neural Information Processing Systems*, volume 38, pages 100773–100804, 2025.

[63] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822, 2023.

[64] Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. LIMO: Less is More for Reasoning. In *Second Conference on Language Modeling*, 2025.

[65] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. In *Advances in Neural Information Processing Systems*, volume 38, pages 113222–113244, 2025.

[66] Weihao Zeng, Keqing He, Chuqiao Kuang, Xiaoguang Li, and Junxian He. Pushing Test-Time Scaling Limits of Deep Search with Asymmetric Verification. In *International Conference on Learning Representations*, pages 115758–115778, 2026.

[67] Di Zhang, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. Accessing GPT-4 level Mathematical Olympiad Solutions via Monte Carlo Tree Self-refine with LLaMa-3 8B. *arXiv preprint arXiv:2406.07394*, 2024.

[68] Tong Zheng, Hongming Zhang, Wenhao Yu, Xiaoyang Wang, He Xing, Runpeng Dai, Rui Liu, Huiwen Bao, Chengsong Huang, Heng Huang, and Dong Yu. Parallel-R1: Towards Parallel Thinking via Reinforcement Learning. In *International Conference on Learning Representations*, pages 121144–121166, 2026.

[69] Peilin Zhou, Bruce Leon, Xiang Ying, Can Zhang, Yifan Shao, Qichen Ye, Dading Chong, Zhiling Jin, Chenxuan Xie, Meng Cao, Yuxin Gu, Sixin Hong, Jing Ren, Jian Chen, Chao Liu, and Yining Hua. BrowseComp-ZH: Benchmarking Web Browsing Ability of Large Language Models in Chinese. *arXiv preprint arXiv:2504.19314*, 2025.

Appendix Contents

A Illustration of Overscaling Curse **17**

B Quantification of Overscaling Curse **20**

C Better Understanding the Overscaling Curse **27**

C.1 Sample Type Classification 27

C.2 Empirical Statistics 31

C.3 Proof of Theorem 1 33

D Detailed Settings of LatBP **34**

D.1 Training Data and Hyperparameters 34

D.2 Inference-time Predictor Weighting 35

E Empirical Validation of Latent Budget Predictor **36**

F Baselines of Efficient Parallel Decoding **38**

G Main Results of Efficient Parallel Decoding **39**

G.1 Supplementary Main Results 39

G.2 Standard Deviations of Main Results 39

G.3 Cost Statistics 40

H Extended Analysis **41**

H.1 Sensitivity to Sampling Hyperparameters 41

H.2 Ablation Study of LatBP 42

H.3 Design Principle of LatBP 44

I Related Work **44**

I.1 Parallel Thinking 44

I.2 Efficient Parallel Decoding 45

I.3 Test-time Scaling 45

I.4 Efficiency Issue in Test-time Scaling 45

A Illustration of Overscaling Curse

Figures 6 – 16 present all illustrations of the overscaling curse across all systems under various aggregation strategies, serving as a complete supplement to Figure 1.

Within each figure, each subfigure corresponds to one answer aggregation strategy. The single thick curve shows how overall dataset accuracy varies with the budget, and thin curves show how sample accuracy varies with the budget for each sample $\boldsymbol{x} \in \mathcal{D}$. We term these curves the *"budget-accuracy" functions*. Hollow markers on each curve indicate the minimum budget that achieves the highest accuracy, which we term the *"optimal budget"*. See Section 2 for experimental setups and Section 3.1 for detailed function computation.

Due to the large data sizes of the GPQA (448 samples), MMLU-Pro (12,000 samples), and BrowseComp (1,255 + 289 samples) datasets, many individual curves would overlap and become unreadable. Therefore, we omit their illustrations here and present only the quantitative results in Section 3.2.

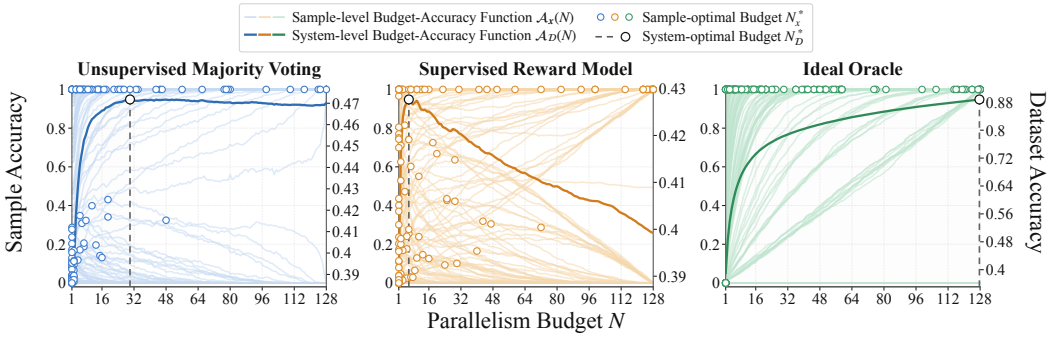


Figure 6: **Illustration of the Overscaling Curse in Parallel Thinking**. Each subfigure corresponds to a system $(\pi, \mathcal{D})$: evaluating model π on dataset $\mathcal{D}$. Here $\pi =$ `Qwen2.5-7B` and $\mathcal{D} =$ `AMC`.

.

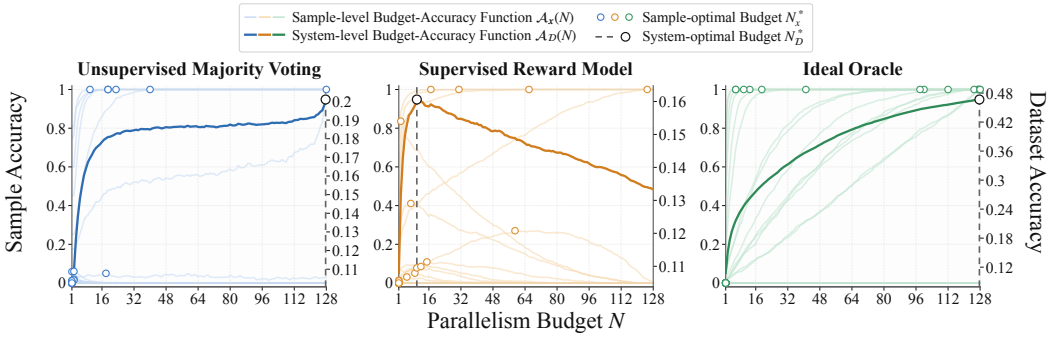


Figure 7: **Illustration of the Overscaling Curse in Parallel Thinking**. Each subfigure corresponds to a system $(\pi, \mathcal{D})$: evaluating model π on dataset $\mathcal{D}$. Here $\pi =$ `Qwen2.5-7B` and $\mathcal{D} =$ `AIME24`.

.

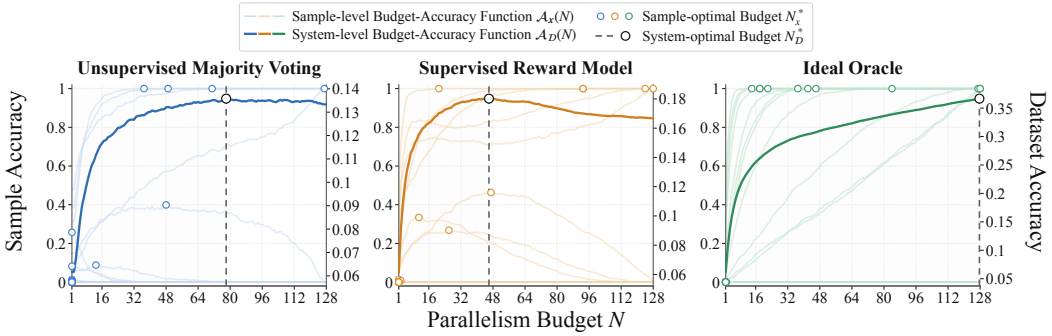


Figure 8: **Illustration of the Overscaling Curse in Parallel Thinking**. Each subfigure corresponds to a system $(\pi, \mathcal{D})$: evaluating model π on dataset $\mathcal{D}$. Here $\pi =$ `Qwen2.5-7B` and $\mathcal{D} =$ `AIME25`.

.

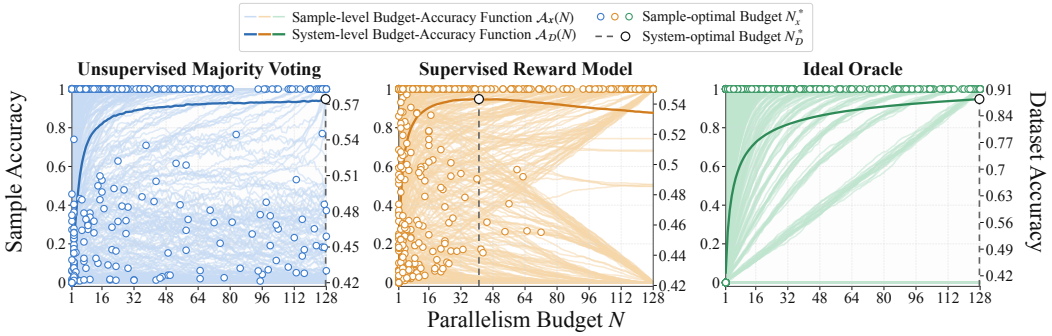


Figure 9: **Illustration of the Overscaling Curse in Parallel Thinking**. Each subfigure corresponds to a system $(\pi, \mathcal{D})$: evaluating model π on dataset $\mathcal{D}$. Here $\pi =$ `Llama3.1-8B` and $\mathcal{D} =$ `MATH-500`.

.

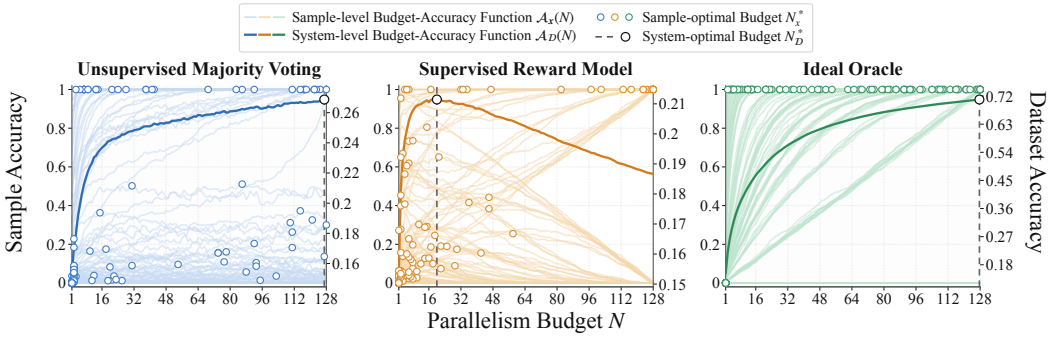


Figure 10: **Illustration of the Overscaling Curse in Parallel Thinking**. Each subfigure corresponds to a system $(\pi, \mathcal{D})$: evaluating model π on dataset $\mathcal{D}$. Here $\pi =$ `Llama3.1-8B` and $\mathcal{D} =$ `AMC`.

.

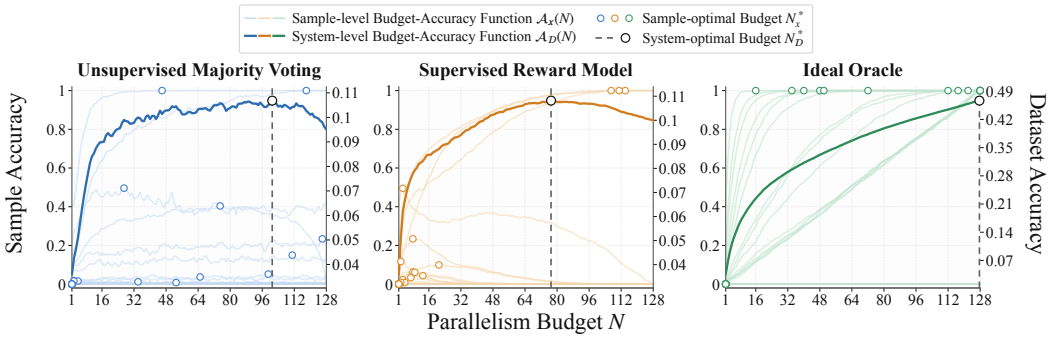


Figure 11: **Illustration of the Overscaling Curse in Parallel Thinking**. Each subfigure corresponds to a system $(\pi, \mathcal{D})$: evaluating model π on dataset $\mathcal{D}$. Here $\pi =$ `Llama3.1-8B` and $\mathcal{D} =$ `AIME24`.

.

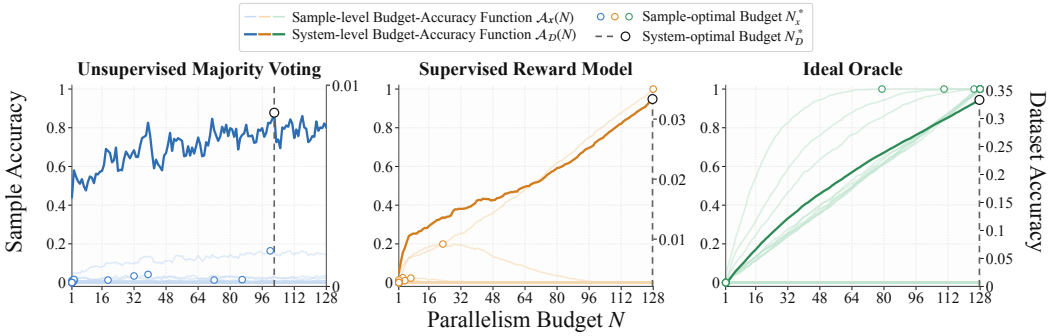


Figure 12: **Illustration of the Overscaling Curse in Parallel Thinking**. Each subfigure corresponds to a system $(\pi, \mathcal{D})$: evaluating model π on dataset $\mathcal{D}$. Here $\pi =$ `Llama3.1-8B` and $\mathcal{D} =$ `AIME25`.

.

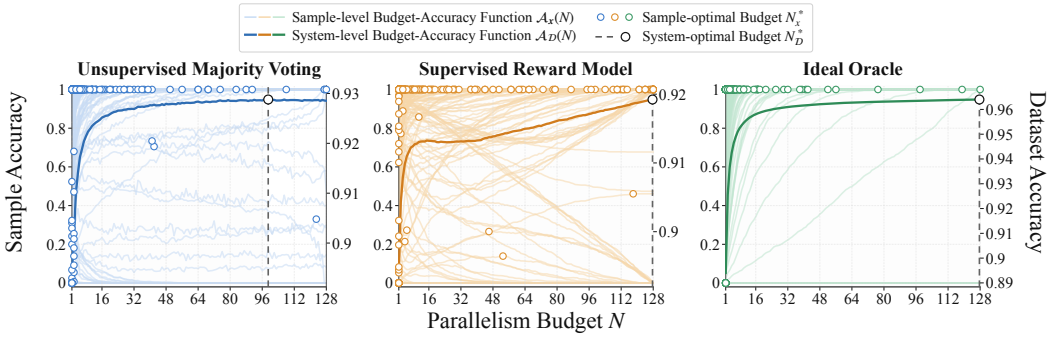


Figure 13: **Illustration of the Overscaling Curse in Parallel Thinking**. Each subfigure corresponds to a system $(\pi, \mathcal{D})$: evaluating model π on dataset $\mathcal{D}$. Here $\pi =$ `Qwen3-4B` and $\mathcal{D} =$ `MATH-500`.

.

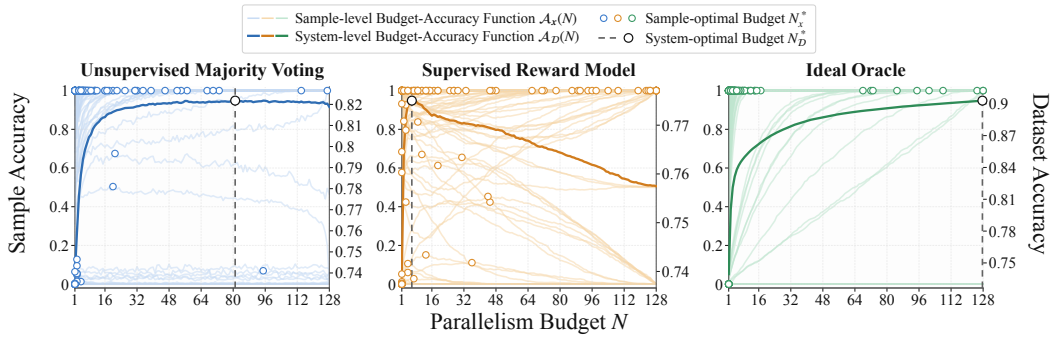


Figure 14: **Illustration of the Overscaling Curse in Parallel Thinking**. Each subfigure corresponds to a system $(\pi, \mathcal{D})$: evaluating model π on dataset $\mathcal{D}$. Here $\pi =$ `Qwen3-4B` and $\mathcal{D} =$ `AMC`.

.

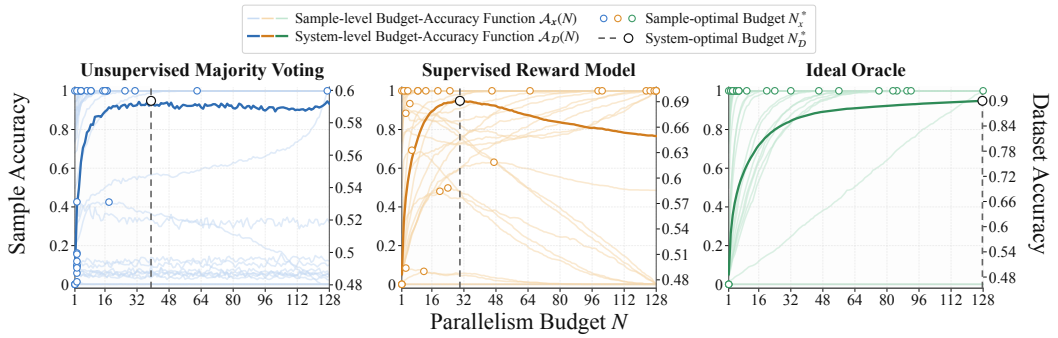


Figure 15: **Illustration of the Overscaling Curse in Parallel Thinking**. Each subfigure corresponds to a system $(\pi, \mathcal{D})$: evaluating model π on dataset $\mathcal{D}$. Here $\pi =$ `Qwen3-4B` and $\mathcal{D} =$ `AIME24`.

.

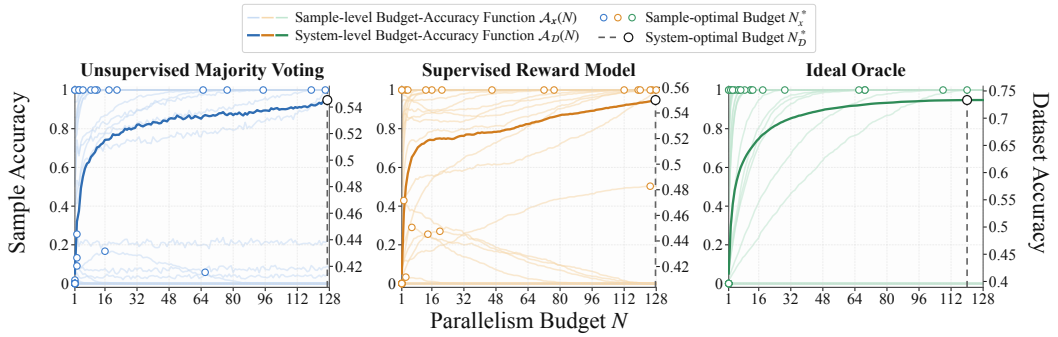


Figure 16: **Illustration of the Overscaling Curse in Parallel Thinking**. Each subfigure corresponds to a system $(\pi, \mathcal{D})$: evaluating model π on dataset $\mathcal{D}$. Here $\pi =$ `Qwen3-4B` and $\mathcal{D} =$ `AIME25`.

.

B Quantification of Overscaling Curse

Figures 17 – 40 present all the distributions of budget utilization and corresponding sample-optimal budgets, serving as a complete supplement to Figure 43.

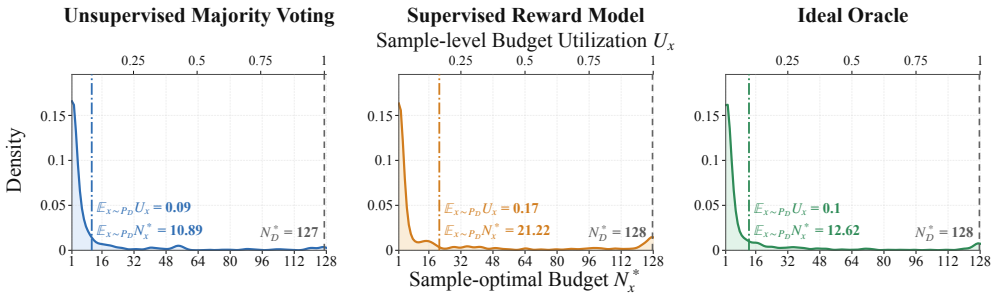


Figure 17: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here π = `Qwen2.5-7B` and $\mathcal{D}$ = `MATH-500`.

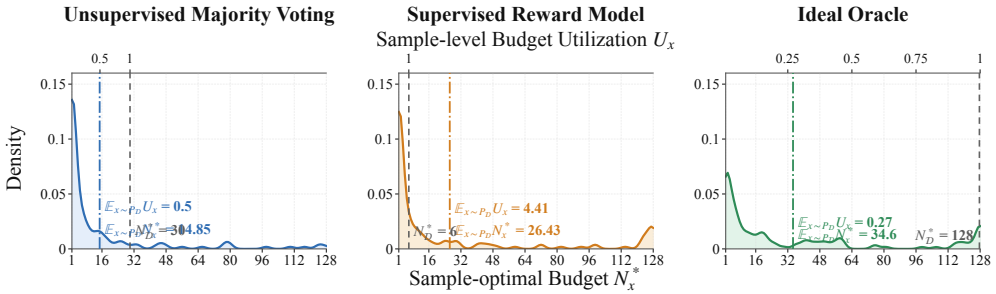


Figure 18: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here π = `Qwen2.5-7B` and $\mathcal{D}$ = `AMC`.

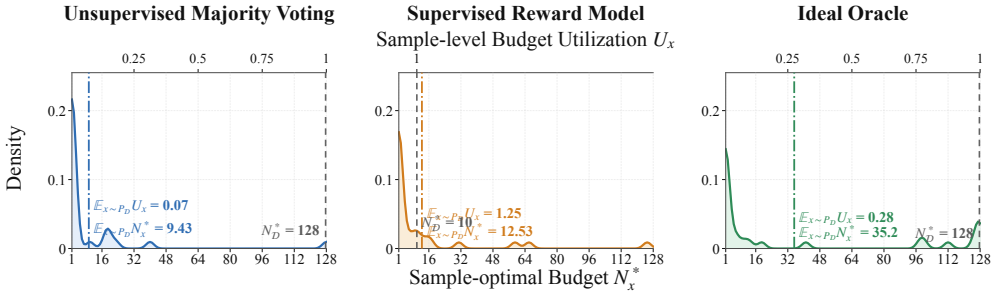


Figure 19: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here π = `Qwen2.5-7B` and $\mathcal{D}$ = `AIME24`.

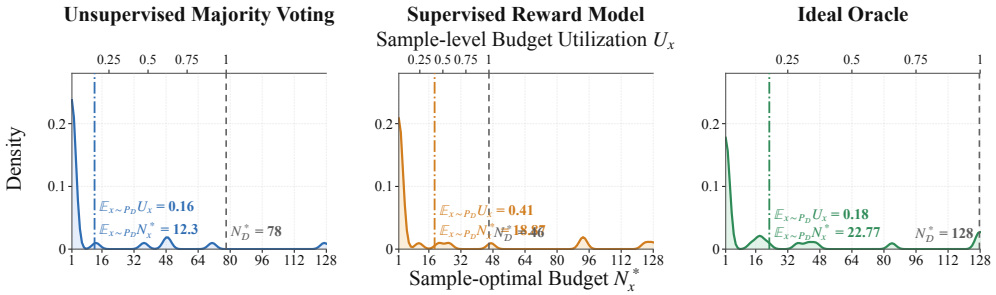


Figure 20: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here π = `Qwen2.5-7B` and $\mathcal{D}$ = `AIME25`.

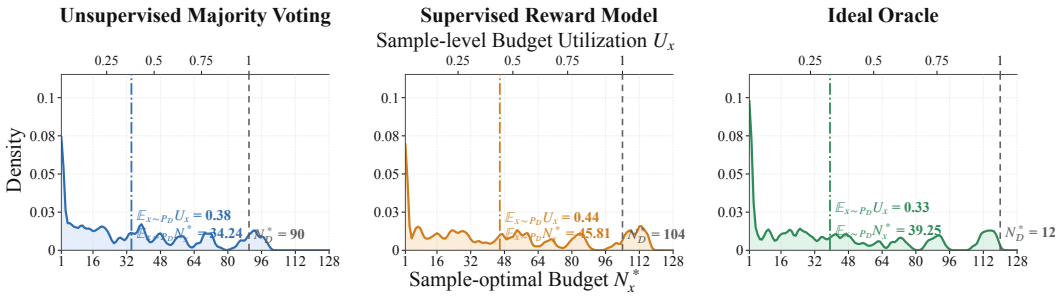


Figure 21: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here $\pi =$ `Qwen2.5-7B` and $\mathcal{D} =$ `GPQA`.

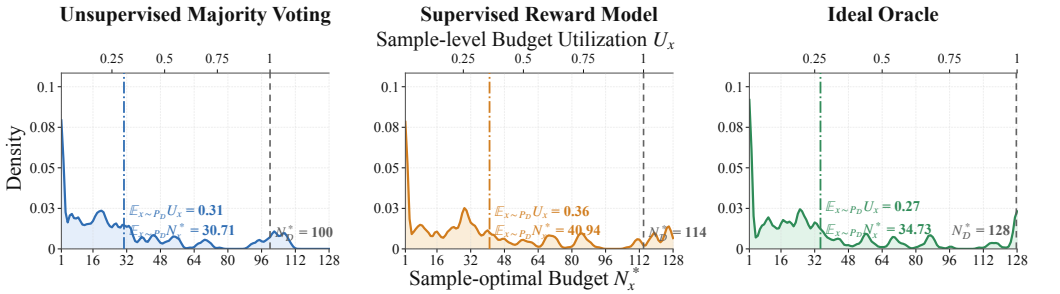


Figure 22: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here $\pi =$ `Qwen2.5-7B` and $\mathcal{D} =$ `MMLU-Pro`.

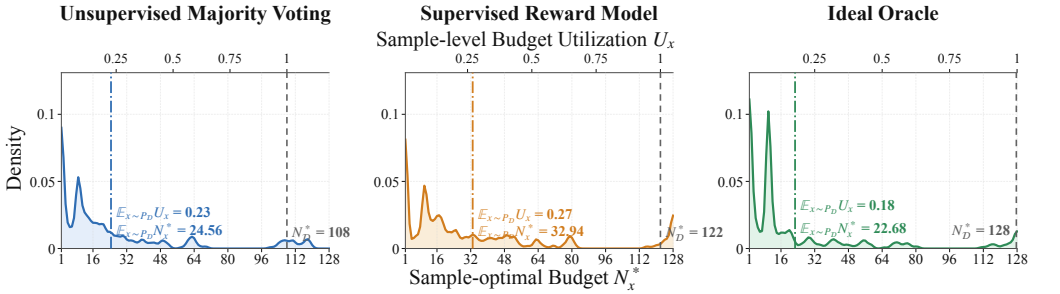


Figure 23: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here $\pi =$ `Qwen2.5-7B` and $\mathcal{D} =$ `BrowseComp`.

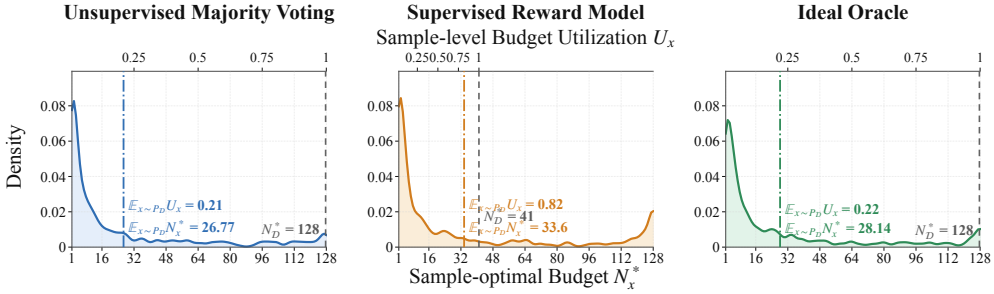


Figure 24: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here $\pi =$ `Llama3.1-8B` and $\mathcal{D} =$ `MATH-500`.

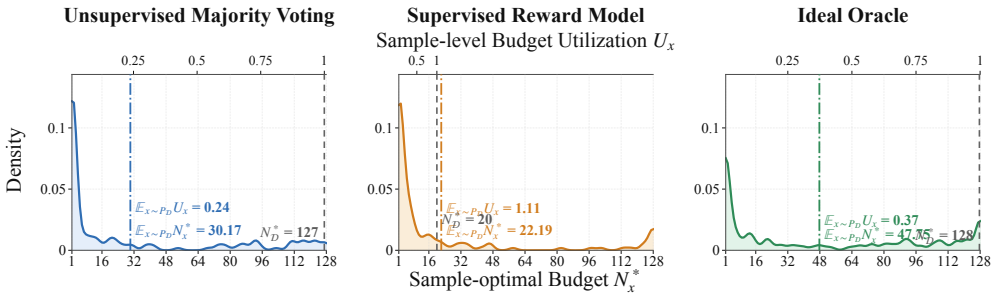


Figure 25: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here $\pi =$ `Llama3.1-8B` and $\mathcal{D} =$ `AMC`.

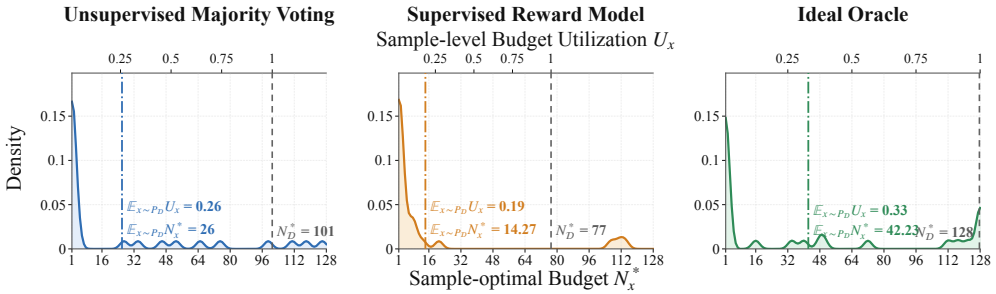


Figure 26: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here $\pi =$ `Llama3.1-8B` and $\mathcal{D} =$ `AIME24`.

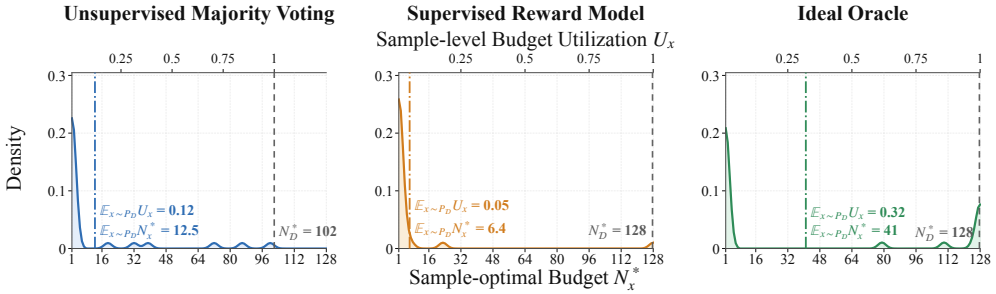


Figure 27: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here $\pi =$ `Llama3.1-8B` and $\mathcal{D} =$ `AIME25`.

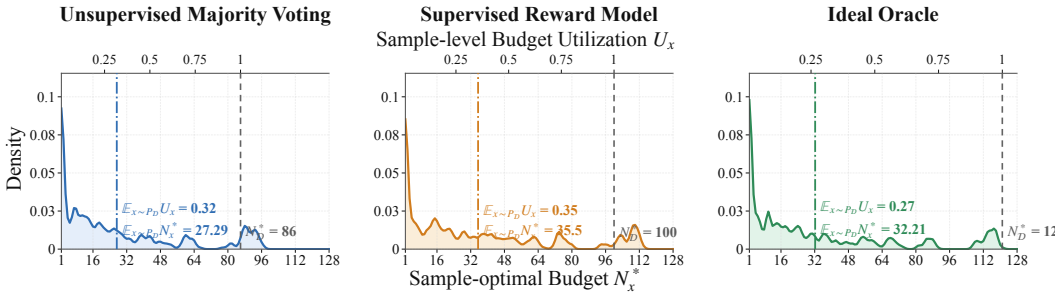


Figure 28: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here $\pi =$ `Llama3.1-8B` and $\mathcal{D} =$ `GPQA`.

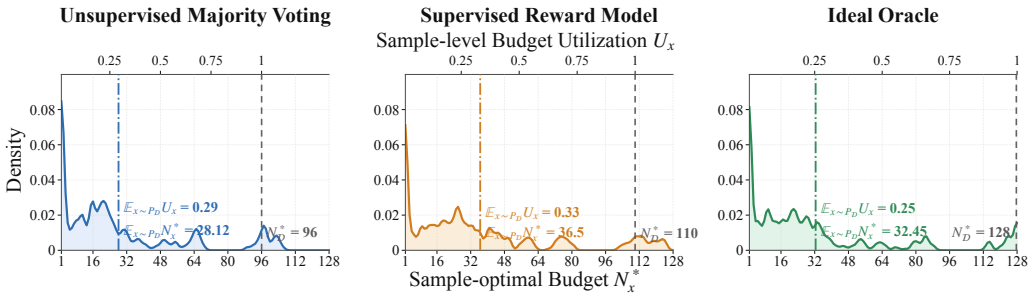


Figure 29: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here $\pi =$ `Llama3.1-8B` and $\mathcal{D} =$ `MMLU-Pro`.

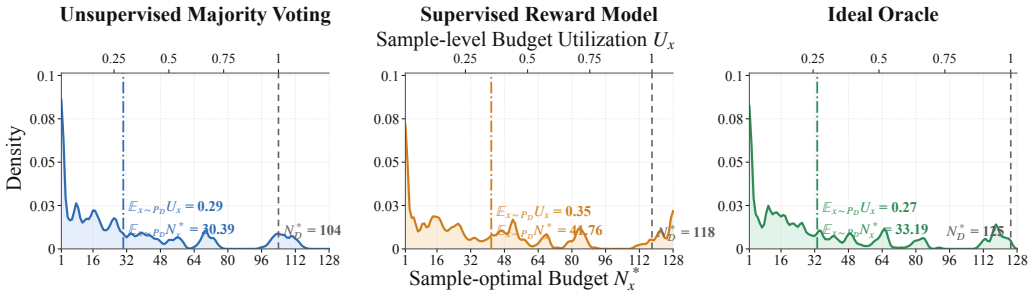


Figure 30: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here $\pi =$ `Llama3.1-8B` and $\mathcal{D} =$ `BrowseComp`.

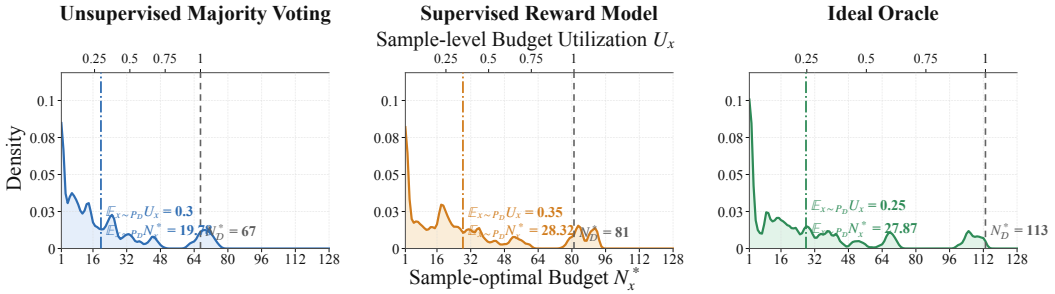


Figure 31: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here $\pi =$ `Deepseek-R1-Distill-Qwen-7B` and $\mathcal{D} =$ `MATH-500`.

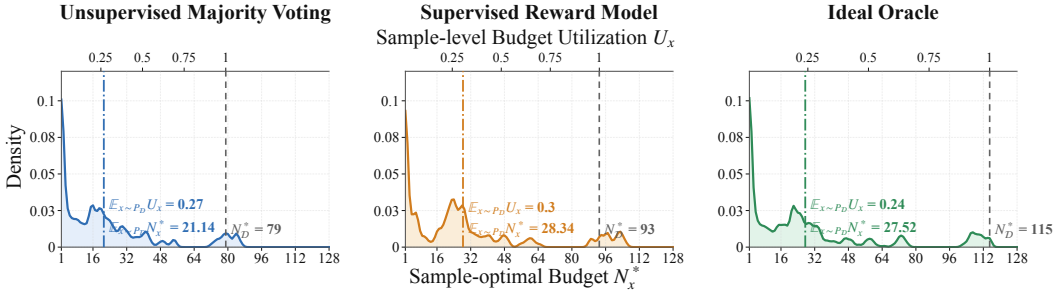


Figure 32: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here $\pi =$ `Deepseek-R1-Distill-Qwen-7B` and $\mathcal{D} =$ `AMC`.

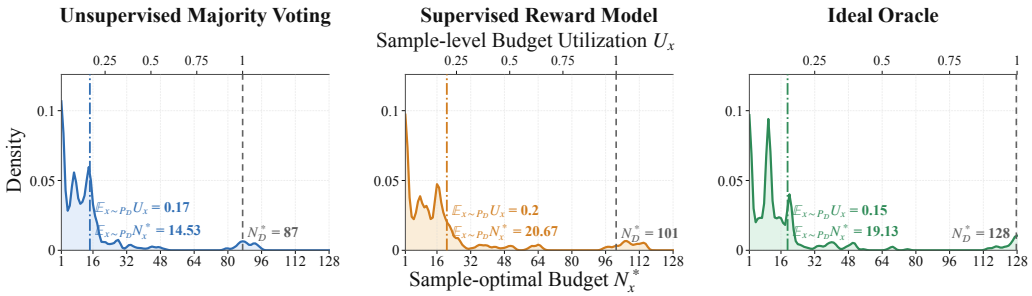


Figure 33: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here $\pi =$ `Deepseek-R1-Distill-Qwen-7B` and $\mathcal{D} =$ `AIME24`.

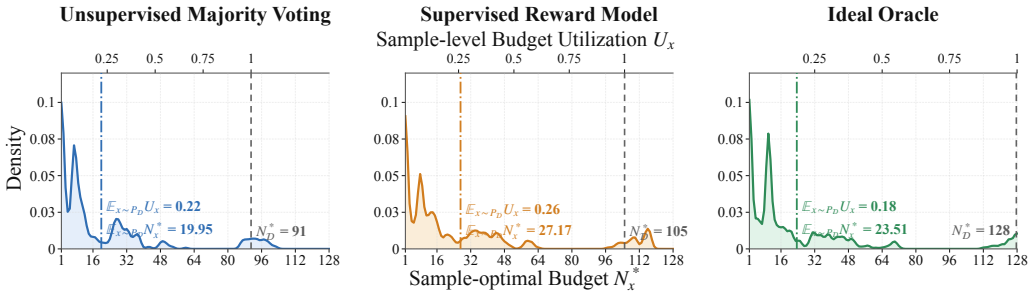


Figure 34: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here $\pi =$ `Deepseek-R1-Distill-Qwen-7B` and $\mathcal{D} =$ `AIME25`.

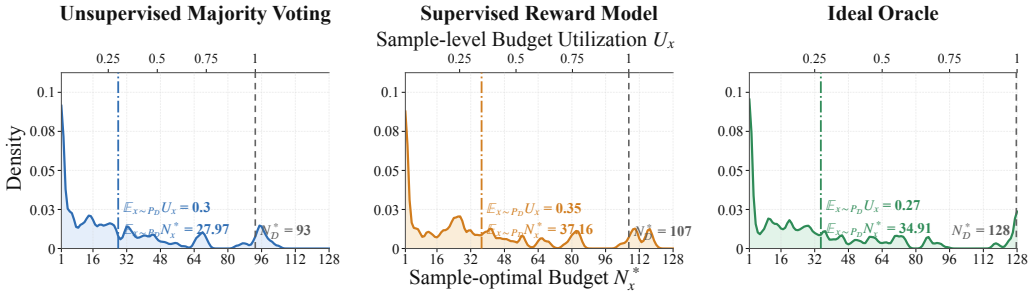


Figure 35: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here $\pi =$ `Deepseek-R1-Distill-Qwen-7B` and $\mathcal{D} =$ `GPQA`.

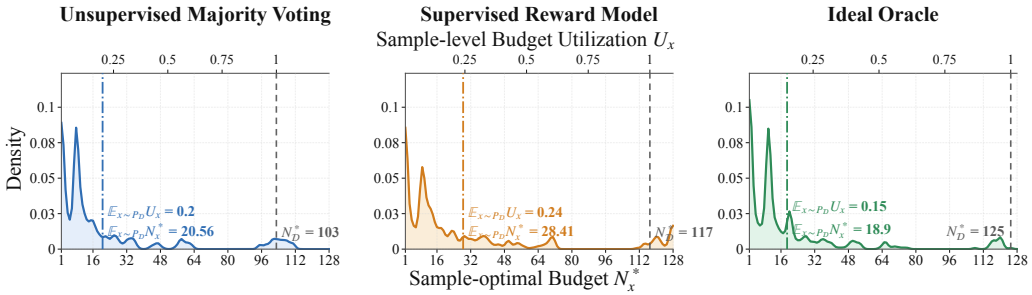


Figure 36: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here $\pi =$ `Deepseek-R1-Distill-Qwen-7B` and $\mathcal{D} =$ `MMLU-Pro`.

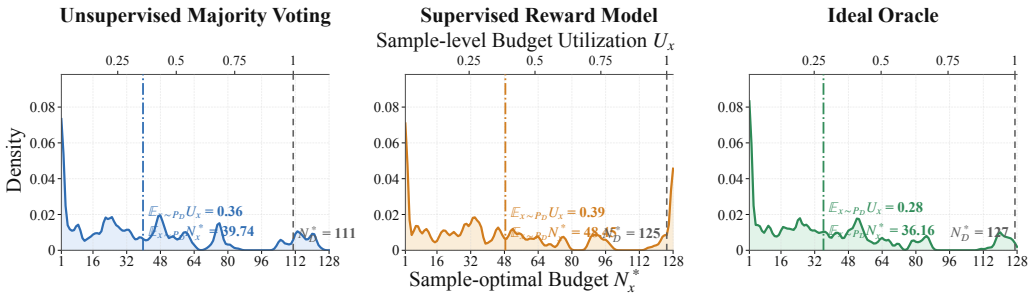


Figure 37: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here $\pi =$ `Deepseek-R1-Distill-Qwen-7B` and $\mathcal{D} =$ `BrowseComp`.

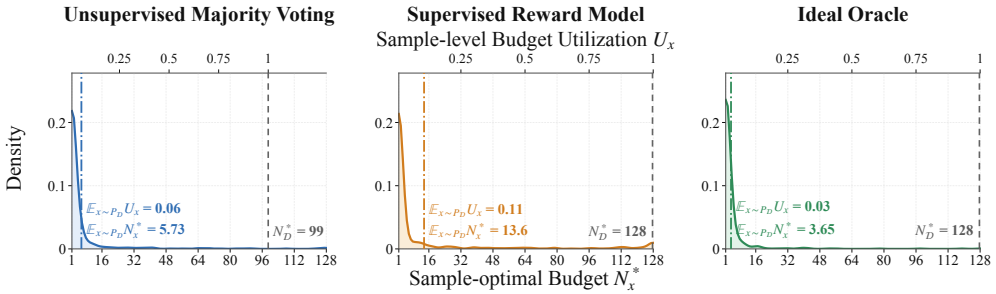


Figure 38: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here $\pi =$ `Qwen3-4B` and $\mathcal{D} =$ `MATH-500`.

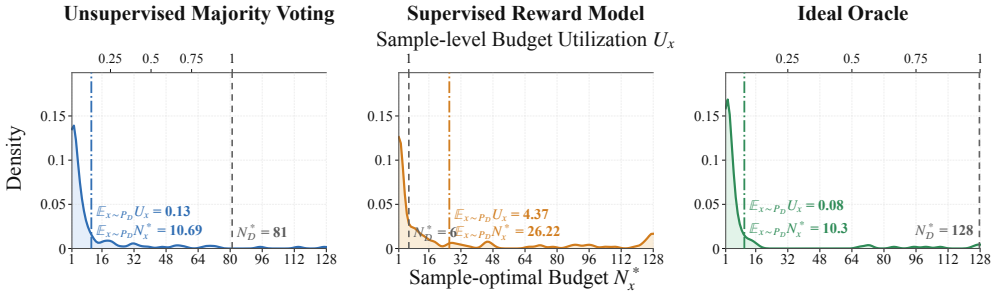


Figure 39: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here $\pi =$ `Qwen3-4B` and $\mathcal{D} =$ `AMC`.

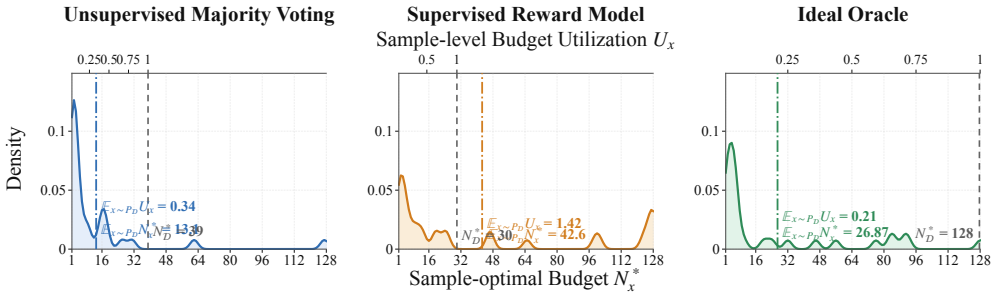


Figure 40: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here $\pi =$ `Qwen3-4B` and $\mathcal{D} =$ `AIME24`.

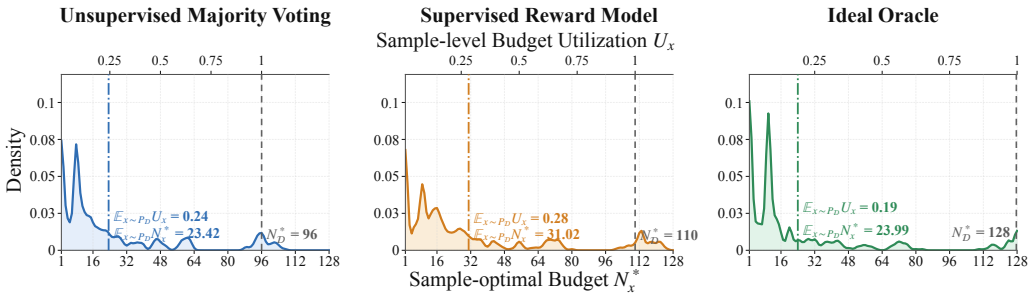


Figure 41: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here π = `Qwen3-4B` and $\mathcal{D}$ = `GPQA`.

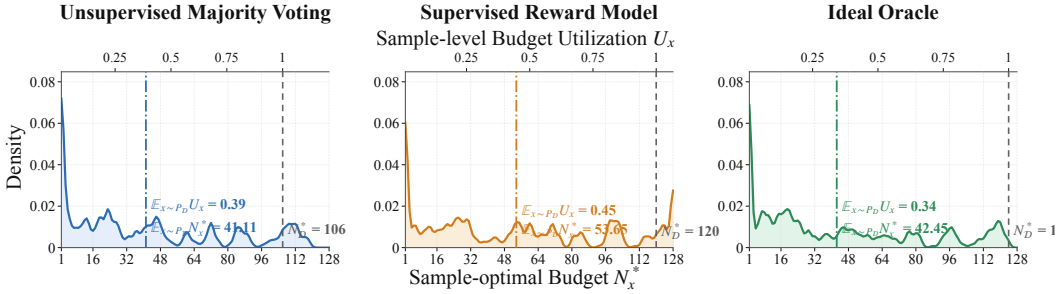


Figure 42: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here π = `Qwen3-4B` and $\mathcal{D}$ = `MMLU-Pro`.

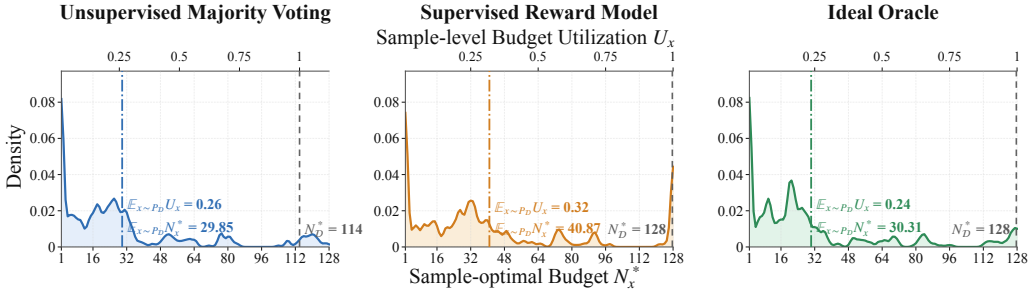


Figure 43: Distribution of *Sample-level Budget Utilization* $\mathcal{U}_{\boldsymbol{x}}$ and corresponding optimal budgets $N_{\boldsymbol{x}}^*$ in a single systems $(\pi, \mathcal{D})$ under different aggregation strategies, along with $N_{\mathcal{D}}^*$. Here π = `Qwen3-4B` and $\mathcal{D}$ = `BrowseComp`.

C Better Understanding the Overscaling Curse

C.1 Sample Type Classification

We classify samples according to the monotonicity pattern of their sample-level budget-accuracy functions $\mathcal{A}_{\boldsymbol{x}}(N)$.

Definition 1 (Approximate Monotonicity of $\mathcal{A}_{\boldsymbol{x}}(N)$) *Let $s := \lfloor \sqrt{N_{\max}} \rfloor$ be the step size for monotonicity checking. For $\alpha \in \{+1, -1\}$, we say that $\mathcal{A}_{\boldsymbol{x}}(N)$ is* approximately monotone *in direction α, denoted by $\mathcal{A}_{\boldsymbol{x}}(N) \approx_\alpha$ mono, if*

$$\frac{1}{N_{\max} - s} \sum_{i=1}^{N_{\max}-s} \mathbb{I}\left\{\alpha \left(\mathcal{A}_{\boldsymbol{x}}(i+s) - \mathcal{A}_{\boldsymbol{x}}(i)\right) \geq 0\right\} \geq 0.80. \tag{8}$$

Here, $\mathcal{A}_{\boldsymbol{x}}(N) \approx_{+1}$ mono denotes approximate increasing monotonicity, while $\mathcal{A}_{\boldsymbol{x}}(N) \approx_{-1}$ mono denotes approximate decreasing monotonicity.

Based on this definition, we partition samples into five types, as summarized in Table 4. Representative examples are shown in Figures 44 – 55.

Table 4: Sample type classification according to the monotonicity pattern of $\mathcal{A}_{\boldsymbol{x}}(N)$. The five subsets form a partition of $\mathcal{D}$, *i.e.*, $\mathcal{D}_i \cap \mathcal{D}_j = \varnothing$ for $i \neq j$ and $\bigcup_{i=1}^{5} \mathcal{D}_i = \mathcal{D}$.

Type ID	Subset	$\mathcal{A}_{\boldsymbol{x}}(N)$ Pattern	Sample Feature
1	$\mathcal{D}_1$	$\mathcal{A}_{\boldsymbol{x}}(N) \equiv 1$	The sample is always solved correctly under all budgets.
2	$\mathcal{D}_2$	$\mathcal{A}_{\boldsymbol{x}}(N) \equiv 0$	The sample is never solved correctly under any budget.
3	$\mathcal{D}_3$	$\mathcal{A}_{\boldsymbol{x}}(N) \approx_{-1}$ mono, with $\mathcal{A}_{\boldsymbol{x}}(N) \not\equiv 1$ and $\mathcal{A}_{\boldsymbol{x}}(N) \not\equiv 0$	Accuracy tends to decrease as the budget increases. This can occur when incorrect answers become more dominant under aggregation.
4	$\mathcal{D}_4$	$\mathcal{A}_{\boldsymbol{x}}(N) \approx_{+1}$ mono, with $\mathcal{A}_{\boldsymbol{x}}(N) \not\equiv 1$ and $\mathcal{A}_{\boldsymbol{x}}(N) \not\equiv 0$	Accuracy tends to increase as the budget increases. This indicates that larger budgets make the correct answer more likely to be selected.
5	$\mathcal{D}_5$	No clear monotonicity	None of the above conditions hold, indicating that the sample does not exhibit a stable monotonic trend over budgets.

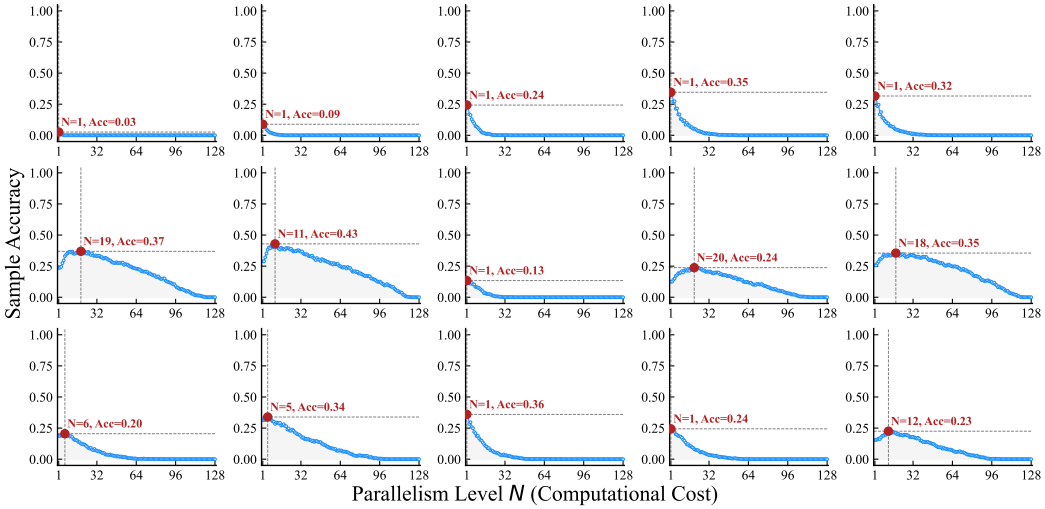


Figure 44: Examples of the "budget-accuracy" function $\mathcal{A}_{\boldsymbol{x}}(N)$ for **Type-(3)** samples from `Qwen2.5-7B` on the `MATH500` dataset.

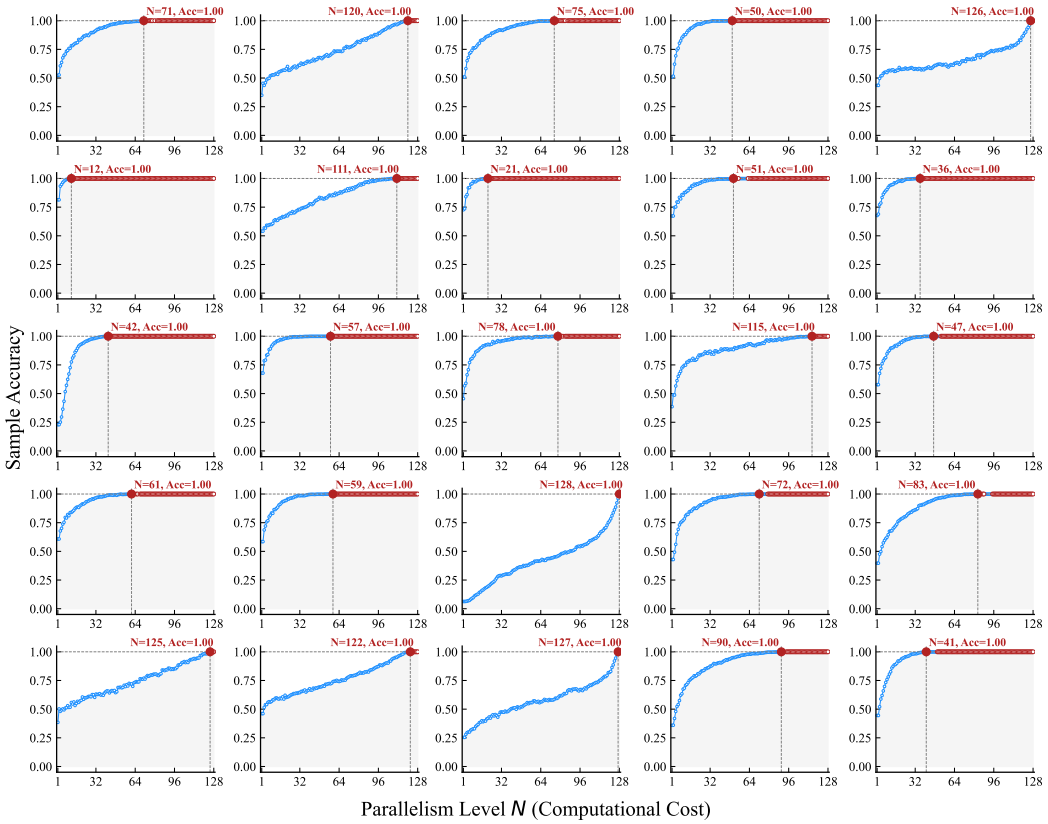


Figure 45: Examples of the "budget-accuracy" function $\mathcal{A}_{\boldsymbol{x}}(N)$ for **Type-(4)** samples from `Qwen2.5-7B` on the `MATH500` dataset.

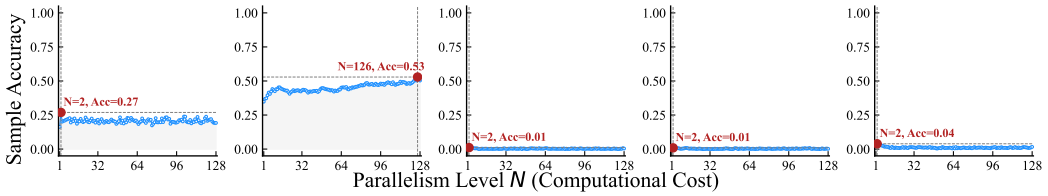


Figure 46: Examples of the "budget-accuracy" function $\mathcal{A}_{\boldsymbol{x}}(N)$ for **Type-(5)** samples from `Qwen2.5-7B` on the `MATH500` dataset.

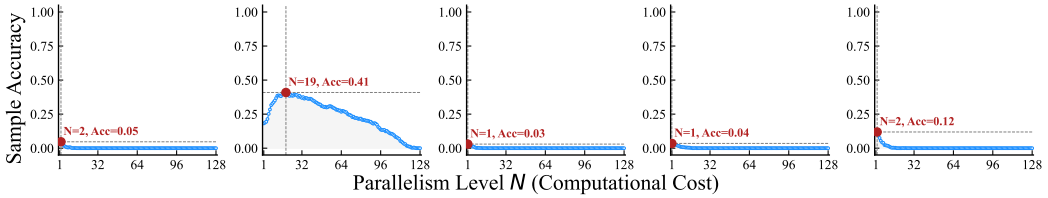


Figure 47: Examples of the "budget-accuracy" function $\mathcal{A}_{\boldsymbol{x}}(N)$ for **Type-(3)** samples from `Qwen2.5-7B` on the `AIME24` dataset.

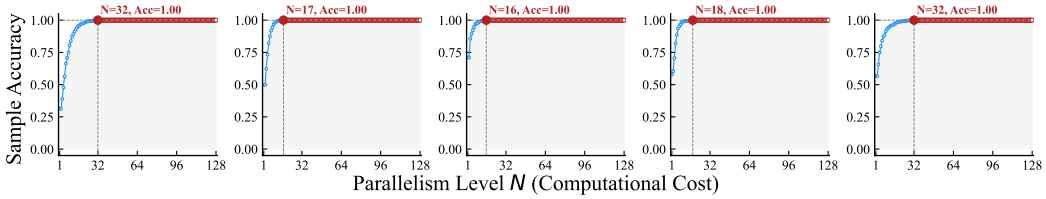


Figure 48: Examples of the "budget-accuracy" function $\mathcal{A}_{\boldsymbol{x}}(N)$ for **Type-(4)** samples from `Qwen2.5-7B` on the `AIME24` dataset.

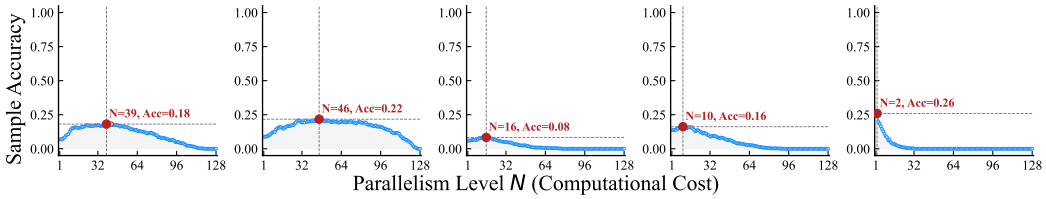


Figure 49: Examples of the "budget-accuracy" function $\mathcal{A}_{\boldsymbol{x}}(N)$ for **Type-(3)** samples from `Qwen2.5-7B` on the `AIME25` dataset.

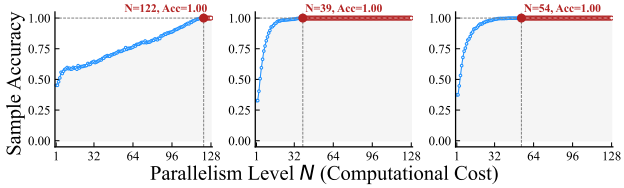


Figure 50: Examples of the "budget-accuracy" function $\mathcal{A}_{\boldsymbol{x}}(N)$ for **Type-(4)** samples from `Qwen2.5-7B` on the `AIME25` dataset.

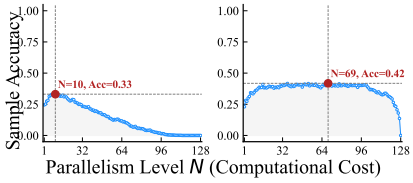


Figure 51: Examples of the "budget-accuracy" function $\mathcal{A}_{\boldsymbol{x}}(N)$ for **Type-(3)** samples from `Qwen3-4B` on the `AIME24` dataset.

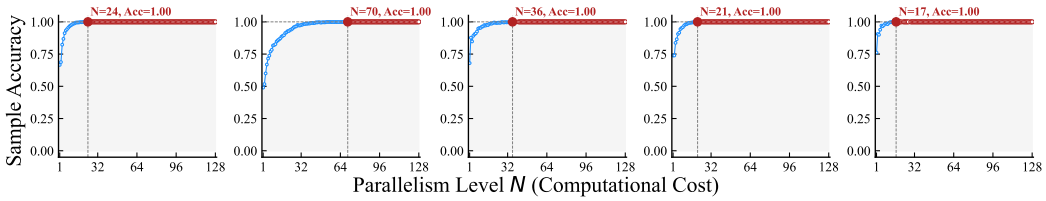


Figure 52: Examples of the "budget-accuracy" function $\mathcal{A}_{\boldsymbol{x}}(N)$ for **Type-(4)** samples from `Qwen3-4B` on the `AIME24` dataset.

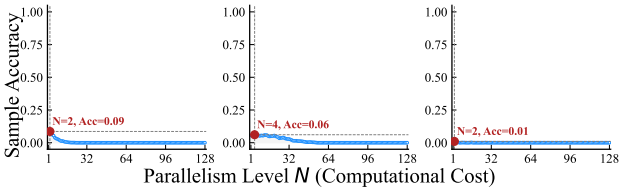


Figure 53: Examples of the "budget-accuracy" function $\mathcal{A}_{\boldsymbol{x}}(N)$ for **Type-(3)** samples from `Qwen3-4B` on the `AIME25` dataset.

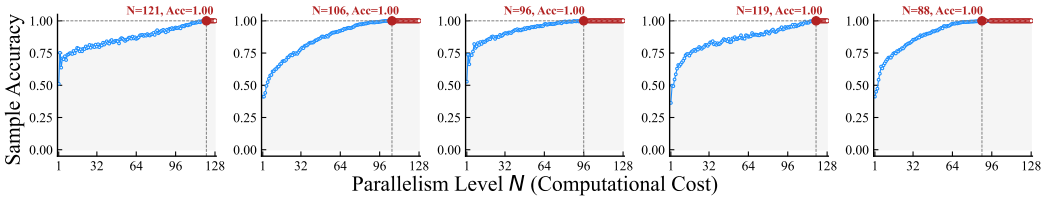


Figure 54: Examples of the "budget-accuracy" function $\mathcal{A}_{\boldsymbol{x}}(N)$ for **Type-(4)** samples from `Qwen3-4B` on the `AIME25` dataset.

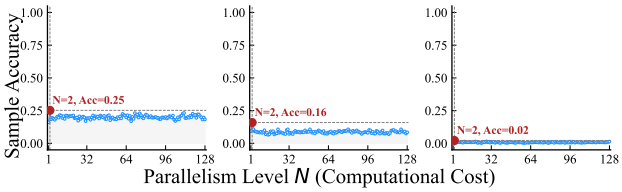


Figure 55: Examples of the "budget-accuracy" function $\mathcal{A}_{\boldsymbol{x}}(N)$ for **Type-(5)** samples from `Qwen3-4B` on the `AIME25` dataset.

C.2 Empirical Statistics

We report the $\mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}_i}}[N^*_{\boldsymbol{x}}]$, p_i, and $\bar{\Delta}_i$ of each sample type subset $\mathcal{D}_i \subset \mathcal{D}$ under all systems with the majority voting aggregation strategy.

Tables 5 – 8 show the $\mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}_i}}[N^*_{\boldsymbol{x}}]$ results. Across systems, Type-4 consistently shows much larger expected $N^*_{\boldsymbol{x}}$ than the other types.

Tables 9 – 12 show the results of p_i, and $\bar{\Delta}_i$. These systems all suffer from severe overscaling curse. Across them, $p_4 > p_3$ and $\bar{\Delta}_4 > [-\bar{\Delta}_3]_+$, where $[z]_+ = \max(z, 0)$, usually happen; p_5 and $\bar{\Delta}_5$ is very small that can negligible.

Table 5: Expected sample-optimal budget $\mathbb{E}[N^*_{\boldsymbol{x}}]$ of each sample type under majority voting. Results are from `Qwen2.5-7B-Instruct`. "–" indicates the absence of such a sample type.

Type	**MATH500**	**AMC**	**AIME24**	**AIME25**	**GPQA**	**MMLU-Pro**	**BrowseComp**
Type-1: Constant-1	1.0	1.0	–	1.0	1.0	1.0	1.0
Type-2: Constant-0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Type-3: Decreasing	3.0	3.4	1.4	3.0	2.4	5.5	6.5
Type-4: Increasing	23.6	33.8	39.8	71.0	43.6	44.5	35.8
Type-5: Non-monotonic	10.9	4.5	18.0	8.0	12.9	11.4	1.4

Table 6: Expected sample-optimal budget $\mathbb{E}[N_{\boldsymbol{x}}^*]$ of each sample type under majority voting. Results are from `Llama-3.1-8B-Instruct`. "–" indicates the absence of such a sample type.

Type	**MATH500**	**AMC**	**AIME24**	**AIME25**	**GPQA**	**MMLU-Pro**	**BrowseComp**
Type-1: Constant-1	1.0	–	–	–	1.0	1.0	1.0
Type-2: Constant-0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Type-3: Decreasing	4.8	3.1	–	1.0	3.5	9.9	1.1
Type-4: Increasing	32.0	59.6	82.0	–	90.2	67.5	35.7
Type-5: Non-monotonic	15.6	44.0	50.0	19.2	35.6	2.2	4.0

Table 7: Expected sample-optimal budget $\mathbb{E}[N_{\boldsymbol{x}}^*]$ of each sample type under majority voting. Results are from `DeepSeek-R1-Distill-Qwen-7B`. "–" indicates the absence of such a sample type.

Type	**MATH500**	**AMC**	**AIME24**	**AIME25**	**GPQA**	**MMLU-Pro**	**BrowseComp**
Type-1: Constant-1	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Type-2: Constant-0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Type-3: Decreasing	6.5	5.1	9.2	7.8	1.3	2.4	3.8
Type-4: Increasing	22.4	75.6	88.2	35.7	48.9	62.3	36.5
Type-5: Non-monotonic	15.3	8.6	57.1	12.8	21.4	13.9	1.2

Table 8: Expected sample-optimal budget $\mathbb{E}[N_{\boldsymbol{x}}^*]$ of each sample type under majority voting. Results are from `Qwen3-4B-Instruct-2507`. "–" indicates the absence of such a sample type.

Type	**MATH500**	**AMC**	**AIME24**	**AIME25**	**GPQA**	**MMLU-Pro**	**BrowseComp**
Type-1: Constant-1	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Type-2: Constant-0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Type-3: Decreasing	1.3	1.3	10.0	6.3	3.4	5.5	2.7
Type-4: Increasing	52.1	17.7	22.0	49.5	55.4	76.1	34.6
Type-5: Non-monotonic	14.2	16.6	2.0	23.3	16.6	12.1	25.7

Table 9: Sample proportion p_i and expected total accuracy gain $\bar{\Delta}_i$ of each sample type on `MATH500` under majority voting. "–" indicates the absence of such a sample type.

Type	**Qwen2.5-7B**		**Llama3.1-8B**		**R1-Distill-Qwen-7B**		**Qwen3-4B**	
	p_i	$\bar{\Delta}_i$	p_i	$\bar{\Delta}_i$	p_i	$\bar{\Delta}_i$	p_i	$\bar{\Delta}_i$
Type-1: Constant-1	0.39	0.00	0.05	0.00	0.56	0.00	0.63	0.00
Type-2: Constant-0	0.08	0.00	0.32	0.00	0.13	0.00	0.04	0.00
Type-3: Decreasing	0.11	-0.12	0.12	-0.14	0.03	-0.04	0.03	-0.08
Type-4: Increasing	0.38	0.21	0.49	0.32	0.28	0.38	0.29	0.22
Type-5: Non-monotonic	0.03	-0.01	0.03	0.05	0.00	0.10	0.02	0.06

Table 10: Sample proportion p_i and expected total accuracy gain $\bar{\Delta}_i$ of each sample type on AMC under majority voting. "–" indicates the absence of such a sample type.

Type	**Qwen2.5-7B**		**Llama3.1-8B**		**R1-Distill-Qwen-7B**		**Qwen3-4B**	
	p_i	$\bar{\Delta}_i$	p_i	$\bar{\Delta}_i$	p_i	$\bar{\Delta}_i$	p_i	$\bar{\Delta}_i$
Type-1: Constant-1	0.09	0.00	–	–	0.46	0.00	0.30	0.00
Type-2: Constant-0	0.11	0.00	0.29	0.00	0.12	0.00	0.10	0.00
Type-3: Decreasing	0.31	-0.10	0.13	-0.05	0.10	-0.04	0.02	-0.04
Type-4: Increasing	0.47	0.32	0.45	0.45	0.30	0.49	0.51	0.19
Type-5: Non-monotonic	0.01	-0.05	0.14	0.03	0.02	0.13	0.07	-0.02

Table 11: Sample proportion p_i and expected total accuracy gain $\bar{\Delta}_i$ of each sample type on AIME24 under majority voting. "–" indicates the absence of such a sample type.

Type	**Qwen2.5-7B**		**Llama3.1-8B**		**R1-Distill-Qwen-7B**		**Qwen3-4B**	
	p_i	$\bar{\Delta}_i$	p_i	$\bar{\Delta}_i$	p_i	$\bar{\Delta}_i$	p_i	$\bar{\Delta}_i$
Type-1: Constant-1	–	–	–	–	0.27	0.00	0.03	0.00
Type-2: Constant-0	0.53	0.00	0.53	0.00	0.20	0.00	0.10	0.00
Type-3: Decreasing	0.20	-0.02	–	–	0.13	-0.07	0.07	-0.11
Type-4: Increasing	0.23	0.49	0.07	0.72	0.37	0.52	0.67	0.27
Type-5: Non-monotonic	0.03	0.00	0.40	0.02	0.03	0.00	0.13	0.00

Table 12: Sample proportion p_i and expected total accuracy gain $\bar{\Delta}_i$ of each sample type on AIME25 under majority voting. "–" indicates the absence of such a sample type.

Type	**Qwen2.5-7B**		**Llama3.1-8B**		**R1-Distill-Qwen-7B**		**Qwen3-4B**	
	p_i	$\bar{\Delta}_i$	p_i	$\bar{\Delta}_i$	p_i	$\bar{\Delta}_i$	p_i	$\bar{\Delta}_i$
Type-1: Constant-1	–	–	–	–	0.20	0.00	0.07	0.00
Type-2: Constant-0	0.63	0.00	0.67	0.00	0.27	0.00	0.27	0.00
Type-3: Decreasing	0.13	-0.07	0.03	-0.01	0.03	-0.05	0.1	-0.06
Type-4: Increasing	0.20	0.68	0.30	0.02	0.33	0.57	0.47	0.33
Type-5: Non-monotonic	0.03	-0.14	0.10	0.00	0.07	0.05	0.10	0.02

C.3 Proof of Theorem 1

Proof. Recall that $\bar{\Delta}_i$ denotes the expected total accuracy gain of $\mathcal{D}_i$ over the full budget domain, i.e.,

$$\bar{\Delta}_i = \mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}_i}}\left[\mathcal{A}_{\boldsymbol{x}}(N_{\max}) - \mathcal{A}_{\boldsymbol{x}}(1)\right]. \tag{9}$$

Let $B_4 = \lceil \mu_4 \rceil$ be the high-budget reference induced by $\mathcal{D}_4$. For the proof, we introduce the interval-wise total gain from N to B_4:

$$\bar{\Delta}_i(N, B_4) = \mathcal{A}_{\mathcal{D}_i}(B_4) - \mathcal{A}_{\mathcal{D}_i}(N), \tag{10}$$

where

$$\mathcal{A}_{\mathcal{D}_i}(N) = \mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}_i}}\left[\mathcal{A}_{\boldsymbol{x}}(N)\right]. \tag{11}$$

Under the monotonic setting, the full-domain dominance condition $p_4\bar{\Delta}_4 > p_3[-\bar{\Delta}_3]_+$ is assumed to transfer to the high-budget interval before B_4:

$$p_4\bar{\Delta}_4(N, B_4) > p_3[-\bar{\Delta}_3(N, B_4)]_+, \qquad \forall N < B_4. \tag{12}$$

This is the interval form of the observed Type-4 dominance: $\mathcal{D}_4$ provides the dominant positive gain, while $\mathcal{D}_3$ provides only a weaker negative pull. Since $\mathcal{D}_1$ and $\mathcal{D}_2$ have zero total gain and $\mathcal{D}_5$ is negligible in practice, the dataset-level accuracy difference between B_4 and any $N < B_4$ can be reduced to

$$\mathcal{A}_{\mathcal{D}}(B_4) - \mathcal{A}_{\mathcal{D}}(N) = p_3\bar{\Delta}_3(N, B_4) + p_4\bar{\Delta}_4(N, B_4). \tag{13}$$

Because

$$p_3\bar{\Delta}_3(N, B_4) \geq -p_3[-\bar{\Delta}_3(N, B_4)]_+, \tag{14}$$

we have

$$p_3\bar{\Delta}_3(N, B_4) + p_4\bar{\Delta}_4(N, B_4) > 0. \tag{15}$$

Therefore,

$$\mathcal{A}_{\mathcal{D}}(B_4) > \mathcal{A}_{\mathcal{D}}(N), \qquad \forall N < B_4. \tag{16}$$

Thus, no budget smaller than B_4 can be system-optimal, implying

$$N^*_{\mathcal{D}} \geq B_4. \tag{17}$$

Next, since $\mu_4 > \mu_i$ for all $i \neq 4$ and $\sum_i p_i = 1$ with $p_4 < 1$, the dataset-level expected sample-optimal budget satisfies

$$\mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}}}[N^*_{\boldsymbol{x}}] = \sum_{i=1}^{5} p_i\mu_i < \mu_4 \leq B_4. \tag{18}$$

Under the global-budget setting, each sample is assigned $N^*_{\mathcal{D}}$, so

$$\mathcal{U}_{\boldsymbol{x}} = \frac{N^*_{\boldsymbol{x}}}{N^*_{\mathcal{D}}}. \tag{19}$$

Taking expectation over $\mathcal{D}$ gives

$$\mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}}}[\mathcal{U}_{\boldsymbol{x}}] = \frac{\mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}}}[N^*_{\boldsymbol{x}}]}{N^*_{\mathcal{D}}} \leq \frac{\mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}}}[N^*_{\boldsymbol{x}}]}{B_4} < 1. \tag{20}$$

This proves the claim. □

D Detailed Settings of LATBP

D.1 Training Data and Hyperparameters

Training/Validation Data. To ensure both difficulty and domain diversity in the training data, we use two sources:

1. For difficulty diversity, we adopt the open-source Deepmath-103K corpus [18], which contains 10 difficulty levels and covers diverse mathematical subdomains. We uniformly sample 250 examples per level, yielding 2,500 examples.
2. For domain diversity, we adopt the open-source WebInstruct-verified corpus [31], which spans nearly 10 domains, including finance, business, history, biology, etc. We uniformly sample from each domain to form another 2,500 examples.

In total, 5,000 examples are used for training. Similarly, the validation data follow the same selection strategy and size, but with no overlap with the training data.

Training Hyperparameters. Table 13 presents the training hyperparameters of all predictors.

Table 13: Training hyperparameters.

Batch Size	128
Epoch	300
Learning Rate	1e-3
Weight Decay	1e-4
Optimizer	AdamW

D.2 Inference-time Predictor Weighting

Solution of the Optimal Weights. In this part, we provide a detailed derivation of the optimal weights. We first restate the problem: During inference, given an unseen sample $\boldsymbol{x}$, let $\hat{\boldsymbol{n}}^*_{\boldsymbol{x}} = (\hat{n}^*_{\boldsymbol{x},1}, \ldots, \hat{n}^*_{\boldsymbol{x},L})^\top$ denote the vector of normalized predictions, and let $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_L)^\top$ be the weight vector satisfying $\mathbf{1}^\top \boldsymbol{\omega} = 1$. The final budget prediction is given by:

$$\hat{N}^*_{\boldsymbol{x}} = N_{\max} \cdot \boldsymbol{\omega}^\top \hat{\boldsymbol{n}}^*_{\boldsymbol{x}}. \tag{21}$$

Our objective is to minimize the expected squared error of the combined prediction on unseen samples:

$$\min_{\boldsymbol{\omega}} \ \mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}}} \left[\left(\boldsymbol{\omega}^\top \hat{\boldsymbol{n}}^*_{\boldsymbol{x}} - n^*_{\boldsymbol{x}} \right)^2 \right] \quad \text{s.t.} \quad \mathbf{1}^\top \boldsymbol{\omega} = 1, \tag{22}$$

where $n^*_{\boldsymbol{x}} = N^*_{\boldsymbol{x}} / N_{\max}$ is the normalized optimal budget.

Solution. Define the error vector $\boldsymbol{e}_{\boldsymbol{x}} = \hat{\boldsymbol{n}}^*_{\boldsymbol{x}} - n^*_{\boldsymbol{x}} \mathbf{1}$. Under the constraint $\mathbf{1}^\top \boldsymbol{\omega} = 1$, the prediction error simplifies to:

$$\boldsymbol{\omega}^\top \hat{\boldsymbol{n}}^*_{\boldsymbol{x}} - n^*_{\boldsymbol{x}} = \boldsymbol{\omega}^\top (\hat{\boldsymbol{n}}^*_{\boldsymbol{x}} - n^*_{\boldsymbol{x}} \mathbf{1}) = \boldsymbol{\omega}^\top \boldsymbol{e}_{\boldsymbol{x}}. \tag{23}$$

Thus, the objective becomes:

$$\mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}}} \left[(\boldsymbol{\omega}^\top \boldsymbol{e}_{\boldsymbol{x}})^2 \right] = \boldsymbol{\omega}^\top \Sigma \boldsymbol{\omega}, \tag{24}$$

where $\Sigma = \mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}}} [\boldsymbol{e}_{\boldsymbol{x}} \boldsymbol{e}_{\boldsymbol{x}}^\top]$ is the error covariance matrix across layers. The optimization problem reduces to:

$$\min_{\boldsymbol{\omega}} \ \boldsymbol{\omega}^\top \Sigma \boldsymbol{\omega} \quad \text{s.t.} \quad \mathbf{1}^\top \boldsymbol{\omega} = 1. \tag{25}$$

We introduce a Lagrange multiplier λ and define the Lagrangian:

$$\mathcal{L}(\boldsymbol{\omega}, \lambda) = \boldsymbol{\omega}^\top \Sigma \boldsymbol{\omega} - \lambda (\mathbf{1}^\top \boldsymbol{\omega} - 1). \tag{26}$$

Taking the derivative with respect to $\boldsymbol{\omega}$ and setting it to zero:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\omega}} = 2\Sigma \boldsymbol{\omega} - \lambda \mathbf{1} = 0 \quad \Rightarrow \quad \boldsymbol{\omega} = \frac{\lambda}{2} \Sigma^{-1} \mathbf{1}. \tag{27}$$

Substituting this into the constraint $\mathbf{1}^\top \boldsymbol{\omega} = 1$:

$$\mathbf{1}^\top \left(\frac{\lambda}{2} \Sigma^{-1} \mathbf{1} \right) = 1 \quad \Rightarrow \quad \frac{\lambda}{2} \cdot \mathbf{1}^\top \Sigma^{-1} \mathbf{1} = 1 \quad \Rightarrow \quad \lambda = \frac{2}{\mathbf{1}^\top \Sigma^{-1} \mathbf{1}}. \tag{28}$$

Plugging λ back into the expression for $\boldsymbol{\omega}$ yields the optimal weights:

$$\boxed{\boldsymbol{\omega}^* = \frac{\Sigma^{-1} \mathbf{1}}{\mathbf{1}^\top \Sigma^{-1} \mathbf{1}}}. \tag{29}$$

□

Approximation Error Analysis. In this part, we analyze the approximation error induced by replacing the full error covariance matrix Σ with its diagonal surrogate. Let $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_L)^\top$ denote the layer-wise prediction errors and $\Sigma = \mathbb{E}[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^\top]$ their covariance matrix. We decompose Σ as

$$\Sigma = D + R, \tag{30}$$

where $D = \operatorname{diag}(\Sigma)$ contains the diagonal elements and $R = \Sigma - D$ contains only the off-diagonal terms. For any aggregation weight $\boldsymbol{w}$ satisfying $\mathbf{1}^\top \boldsymbol{w} = 1$ and $w_l \geq 0$, the resulting mean squared error can be written as

$$\boldsymbol{w}^\top \Sigma \boldsymbol{w} = \boldsymbol{w}^\top D \boldsymbol{w} + \boldsymbol{w}^\top R \boldsymbol{w}. \tag{31}$$

The second term quantifies the approximation error introduced by ignoring cross-layer correlations. By Hölder's inequality, we have

$$|\boldsymbol{w}^\top R \boldsymbol{w}| \leq \|R\|_\infty := \max_i \sum_j |R_{ij}|, \tag{32}$$

which yields the bound

$$\boldsymbol{w}^\top \Sigma \boldsymbol{w} \in \Big[\boldsymbol{w}^\top D\boldsymbol{w} - \|R\|_\infty,\ \boldsymbol{w}^\top D\boldsymbol{w} + \|R\|_\infty\Big]. \tag{33}$$

Therefore, the worst-case deviation of the diagonal surrogate from the true aggregation error is upper bounded by $\|R\|_\infty$. In practice, we estimate the empirical covariance matrix $\hat{\Sigma}$ on the validation set and report the relative off-diagonal energy

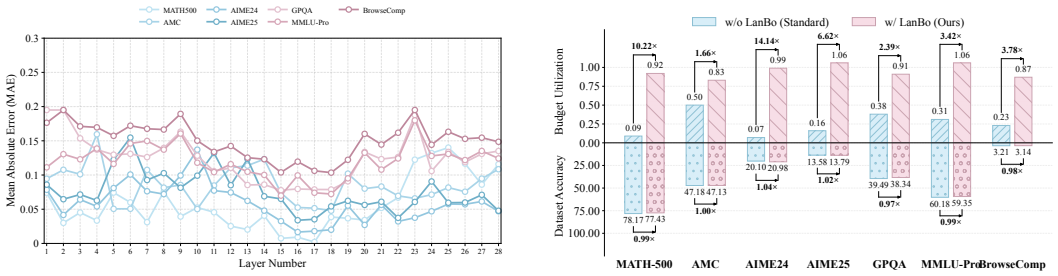
$$\frac{\|\hat{R}\|_F}{\|\hat{\Sigma}\|_F}, \qquad \hat{R} = \hat{\Sigma} - \mathrm{diag}(\hat{\Sigma}), \tag{34}$$

as well as the relative MSE deviation

$$\frac{|\boldsymbol{w}^\top \hat{R} \boldsymbol{w}|}{\boldsymbol{w}^\top \hat{D} \boldsymbol{w}}. \tag{35}$$

Empirically, both quantities are consistently small, indicating that cross-layer error correlations contribute marginally to the aggregated MSE, and validating the diagonal approximation adopted in our method.

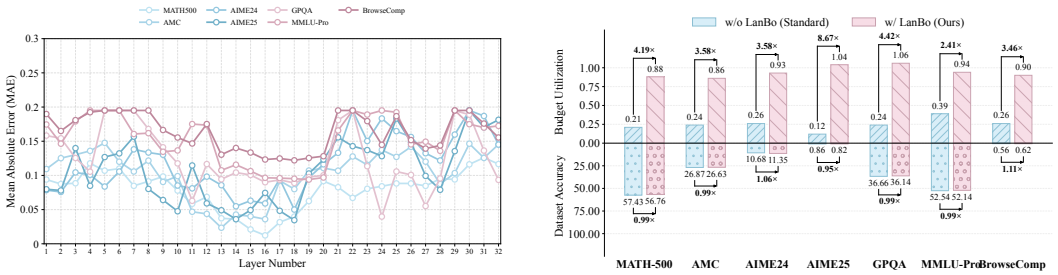
E Empirical Validation of Latent Budget Predictor



(a) MAE results of the L layer-wise predictors on the validation set, with each averaged *over 8 runs*.

(b) Comparison of system-level budget utilization and dataset accuracy before and after applying LATBP.

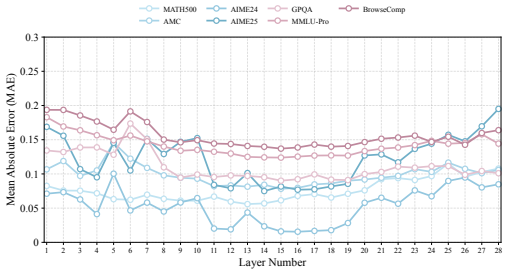
Figure 56: **Empirical Validation of LATBP for Breaking the Overscaling Curse**. Here we use `Qwen2.5-7B` with *voting* aggregation.



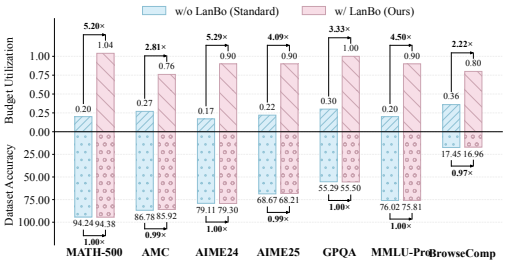
(a) MAE results of the L layer-wise predictors on the validation set, with each averaged *over 8 runs*.

(b) Comparison of system-level budget utilization and dataset accuracy before and after applying LATBP.

Figure 57: **Empirical Validation of LATBP for Breaking the Overscaling Curse**. Here we use `Llama3.1-8B` with *voting* aggregation.

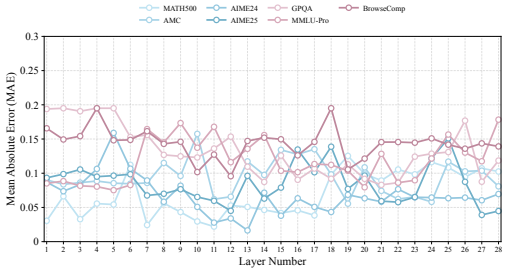


(a) MAE results of the L layer-wise predictors on the validation set, with each averaged *over 8 runs*.

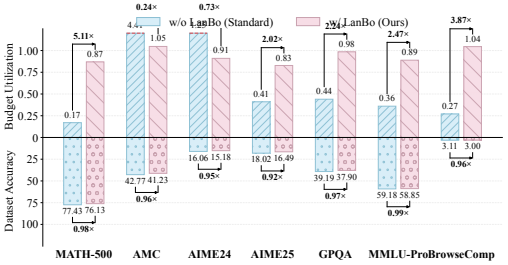


(b) Comparison of system-level budget utilization and dataset accuracy before and after applying LATBP.

Figure 58: **Empirical Validation of LATBP for Breaking the Overscaling Curse**. Here we use `Deepseek-R1-Distill-Qwen-7B` with *voting* aggregation.

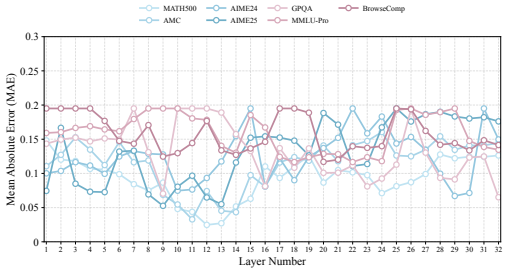


(a) MAE results of the L layer-wise predictors on the validation set, with each averaged *over 8 runs*.

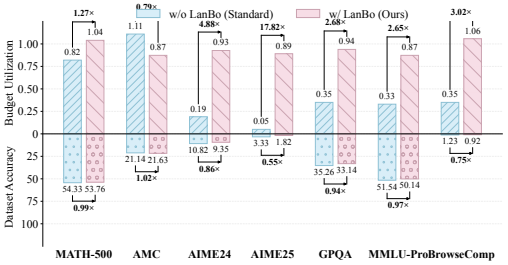


(b) Comparison of system-level budget utilization and dataset accuracy before and after applying LATBP.

Figure 59: **Empirical Validation of LATBP for Breaking the Overscaling Curse**. Here we use `Qwen2.5-7B` with *reward model* aggregation.

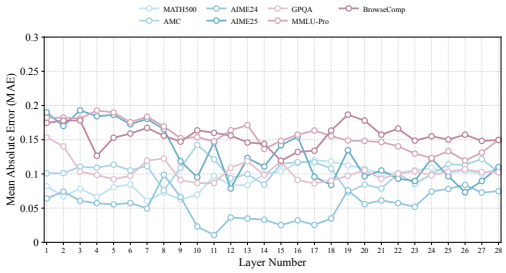


(a) MAE results of the L layer-wise predictors on the validation set, with each averaged *over 8 runs*.

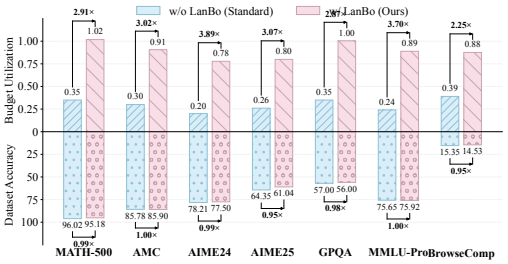


(b) Comparison of system-level budget utilization and dataset accuracy before and after applying LATBP.

Figure 60: **Empirical Validation of LATBP for Breaking the Overscaling Curse**. Here we use `Llama3.1-8B` with *reward model* aggregation.

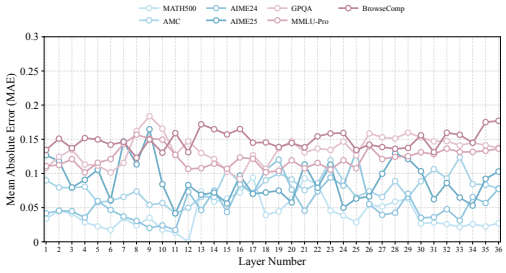


(a) MAE results of the L layer-wise predictors on the validation set, with each averaged *over 8 runs*.

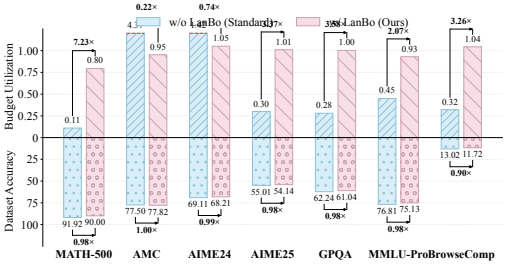


(b) Comparison of system-level budget utilization and dataset accuracy before and after applying LATBP.

Figure 61: **Empirical Validation of LATBP for Breaking the Overscaling Curse**. Here we use `Deepseek-R1-Distill-Qwen-7B` with *reward model* aggregation.



(a) MAE results of the L layer-wise predictors on the validation set, with each averaged *over 8 runs*.



(b) Comparison of system-level budget utilization and dataset accuracy before and after applying LATBP.

Figure 62: **Empirical Validation of LATBP for Breaking the Overscaling Curse**. Here we use `Qwen3-4B` with *reward model* aggregation.

F Baselines of Efficient Parallel Decoding

AC [1]. Adaptive-Consistency (AC) introduces a Beta Stopping Criteria. For each input, **AC samples outputs one by one** and performs an early-stopping check after each sampling. Suppose that n outputs have been sampled so far, yielding m distinct results with an distribution $(p_1, p_2, \ldots, p_m)$, where $p_1 \geq p_2 \geq \cdots \geq p_m$. If the condition $P(p_1 > p_2) \approx \int_0^{0.5} p_2^{v_2} \cdot (1-p_2)^{v_1} \, \mathrm{d}p_2 > 0.95$ is satisfied, sampling stops and the result corresponding to p_1 is returned as the final answer; otherwise, sampling continues until the maximum budget L is reached. In the original paper, $L = 40$, and we adopt the same settings in our implementation.

ESC [26]. Early-Stopping Self-Consistency (ESC) adopts a sliding-window entropy mechanism for adaptive early stopping. Specifically, for each input, **ESC samples w outputs in parallel at each step, where w denotes the window size**. If all outputs within the current window are identical, sampling terminates and this output is returned as the final answer. Otherwise, ESC continues to sample another w outputs until reaching the maximum budget L, after which the most frequent output is selected as the final answer. In the original paper, $w = 8$ and $L = 64$ are used for MATH500, while $w = 5$ and $L = 40$ are used for other datasets. We adopt the same settings in our implementation.

DSC [48]. Difficulty-Adaptive Self-Consistency (DSC) follows a three-stage procedure. In Stage 1, the model ranks questions by estimating difficulty. In Stage 2, it processes questions from hardest to easiest and draws w samples per question. If k consecutive questions are encountered for which all w samples agree, it stops this pass and sets the sampling budget of all remaining (easier) questions to 1. In Stage 3, using the stopping point as a threshold, DSC draws one sample for each easier question, while for harder questions it adaptively increases the budget by doubling the number of w-sample blocks, up to a maximum of L samples. In the original paper, $w = 4$, $k = 32$, and $L = 40$. We adopt the same settings in our implementation.

DeepConf [12]. Deep Think with Confidence (DeepConf) includes both offline and online algorithms; we mainly compare with its **online** variant, which belongs to efficient paradigms. The online algorithm consists of two stages. First, an *offline warm-up* stage **samples N_{init} outputs in parallel** to compute a trace confidence threshold s. Then, in the *adaptive sampling* stage, **outputs are sampled one by one**. During each sampling, a confidence check is performed every n tokens: if the trace confidence within a window falls below s, the current trace is truncated and a new sampling begins; otherwise, sampling continues until completion. After completing this sampling, trace-confidence-weighted majority voting is conducted over all collected outputs so for. If the highest frequency exceeds a threshold τ, sampling stops and that output is returned as the final answer; otherwise, sampling continues until reaching the maximum budget L. We set $N_{\text{init}} = 16$, $\tau = 0.95$, and $L = 512$ as in the original paper.

G Main Results of Efficient Parallel Decoding

G.1 Supplementary Main Results

Tables 14 – 16 presents empirical validation of our LatBP for efficient parallel decoding, supplementing the `Qwen3-4B` results shown in Table **??** in the main text.

Under each system, we report the real-world efficiency (memory $\delta_{\mathcal{M}}$ and latency $\delta_{\mathcal{T}}$, both the ratio relative to *STD.*) and accuracy (*Acc.*), where *STD.* denotes the standard parallel decoding. **Bold** and underline denote the best and second best, respectively. Blue means higher accuracy than *STD.*; red means lower efficiency. Each result is averaged ***over 32 runs***.

Table 14: **Empirical Validation of Our LatBP for Efficient Parallel Decoding.** This table shows `Qwen2.5-7B` results.

Paradigm	Method	MATH500			AMC			AIME24			AIME25			GPQA			MMLU-Pro			BrowseComp		
		$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑	$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑	$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑	$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑	$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑	$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑	$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑
—	*STD.*	*1.00*	*1.00*	*78.17*	*1.00*	*1.00*	*47.18*	*1.00*	*1.00*	*20.10*	*1.00*	*1.00*	*13.58*	*1.00*	*1.00*	*39.49*	*1.00*	*1.00*	*60.18*	*1.00*	*1.00*	*3.21*
InAda	AC	0.24	1.49	77.26	0.28	2.32	46.67	0.74	3.46	19.65	0.66	2.83	11.94	0.38	2.70	38.79	0.32	2.03	59.28	0.40	1.76	2.58
	ESC	0.52	2.77	77.58	0.34	1.90	46.77	0.32	0.96	19.72	0.33	1.45	12.02	0.36	2.27	39.05	0.28	1.78	59.38	0.35	1.65	2.71
	DSC	0.44	1.43	77.33	0.31	2.16	46.85	**0.27**	1.83	18.52	**0.28**	0.97	9.89	0.31	2.18	38.12	0.40	3.74	59.59	0.41	3.12	1.92
	DeepConf	0.45	3.45	**77.82**	0.38	2.67	46.67	1.13	3.16	18.08	0.78	1.88	12.25	0.53	3.41	**39.19**	0.61	3.21	59.63	0.65	2.18	3.00
PreAda	LatBP (Ours)	**0.18**	**0.48**	77.49	**0.23**	**0.74**	**47.08**	0.49	**0.52**	**20.91**	0.35	**0.68**	**13.84**	**0.27**	**0.51**	38.32	**0.23**	**0.39**	**59.85**	**0.30**	**0.49**	**3.17**

Table 15: **Empirical Validation of Our LatBP for Efficient Parallel Decoding.** This table shows `Llama3.1-8B` results.

Paradigm	Method	MATH500			AMC			AIME24			AIME25			GPQA			MMLU-Pro			BrowseComp		
		$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑	$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑	$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑	$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑	$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑	$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑	$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑
—	*STD.*	*1.00*	*1.00*	*57.43*	*1.00*	*1.00*	*26.87*	*1.00*	*1.00*	*10.68*	*1.00*	*1.00*	*0.86*	*1.00*	*1.00*	*36.66*	*1.00*	*1.00*	*52.54*	*1.00*	*1.00*	*0.56*
InAda	AC	**0.18**	1.71	55.90	0.21	2.27	25.73	0.34	3.56	9.85	0.41	3.50	0.78	0.37	2.99	36.24	0.31	3.17	52.58	0.34	2.12	0.51
	ESC	0.34	1.40	**56.65**	0.53	2.03	25.56	0.33	1.64	11.28	0.21	2.44	0.69	0.32	2.38	36.45	0.35	2.16	**52.63**	0.50	1.88	0.50
	DSC	0.28	1.75	55.38	0.33	2.18	25.56	0.27	1.63	7.98	**0.14**	0.98	0.35	0.26	2.08	35.92	0.26	2.83	52.00	0.35	2.44	0.43
	DeepConf	0.44	2.37	56.12	0.48	2.61	25.82	0.67	4.02	7.43	0.72	5.22	0.47	0.52	4.17	34.49	0.63	4.16	49.86	0.66	3.19	0.44
PreAda	LatBP (Ours)	0.23	**0.41**	56.62	**0.16**	**0.37**	**26.69**	**0.22**	**0.43**	**11.39**	0.17	**0.52**	**0.80**	**0.25**	**0.37**	**36.48**	**0.22**	**0.42**	52.23	**0.26**	**0.45**	**0.62**

Table 16: **Empirical Validation of Our LatBP for Efficient Parallel Decoding.** This table shows `Deepseek-R1-Distill-Qwen-7B` results.

Paradigm	Method	MATH500			AMC			AIME24			AIME25			GPQA			MMLU-Pro			BrowseComp		
		$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑	$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑	$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑	$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑	$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑	$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑	$\delta_{\mathcal{M}}\downarrow$	$\delta_{\mathcal{T}}\downarrow$	*Acc.*↑
—	*STD.*	*1.00*	*1.00*	*94.24*	*1.00*	*1.00*	*86.78*	*1.00*	*1.00*	*79.11*	*1.00*	*1.00*	*68.67*	*1.00*	*1.00*	*55.29*	*1.00*	*1.00*	*76.02*	*1.00*	*1.00*	*17.45*
InAda	AC	0.28	1.89	93.70	0.32	2.77	85.82	0.44	2.50	78.64	0.37	2.75	67.79	0.54	2.62	54.67	0.47	2.32	75.72	**0.28**	1.76	16.83
	ESC	**0.15**	0.84	94.02	0.27	1.72	**86.03**	0.53	2.19	78.39	0.41	2.19	68.12	0.43	1.67	54.42	0.38	1.92	75.37	0.45	1.92	14.23
	DSC	0.45	2.13	91.04	0.39	2.89	81.28	0.56	3.58	73.75	0.32	2.00	65.30	0.60	4.03	52.99	0.35	1.79	71.14	0.46	2.11	15.23
	DeepConf	0.53	3.86	93.45	0.58	3.56	85.52	0.83	4.98	**79.58**	0.75	3.57	**68.90**	0.65	3.80	53.13	0.59	3.88	73.40	0.66	3.12	16.09
PreAda	LatBP (Ours)	0.19	**0.37**	**94.32**	**0.22**	**0.48**	85.97	**0.34**	**0.71**	79.30	**0.26**	**0.50**	68.28	**0.25**	**0.45**	**55.53**	**0.23**	**0.49**	**75.84**	0.31	**0.52**	**16.96**

G.2 Standard Deviations of Main Results

Table 17 reports the standard deviations of accuracy (*Acc.*) of all methods under each system in Table **??** and 14 – 16.

Table 17: Standard deviation of accuracy (*Acc.*) corresponding to Table **??** and 14 – 16 across 32 runs.

Paradigm	Method	MATH500	AMC	AIME24	AIME25	GPQA	MMLU-Pro	BrowseComp
				`Qwen2.5-7B`				
—	*STD.*	0.32	0.41	0.28	0.14	0.55	0.49	0.08
INADA	AC	0.33	0.51	0.39	0.12	0.74	0.66	0.05
	ESC	0.38	0.25	0.26	0.13	0.61	0.59	0.09
	DSC	0.31	0.49	0.23	0.29	0.70	0.68	0.02
	DeepConf	0.37	0.44	0.22	0.26	0.88	0.91	0.08
PREADA	**LATBP (Ours)**	0.31	0.35	0.44	0.11	0.37	0.33	0.07
				`Llama3.1-8B`				
—	*STD.*	0.46	0.33	0.12	0.19	0.57	0.52	0.01
INADA	AC	0.68	0.46	0.25	0.11	0.79	0.64	0.06
	ESC	0.61	0.40	0.14	0.11	0.66	0.63	0.03
	DSC	0.65	0.33	0.18	0.15	0.71	0.69	0.06
	DeepConf	0.82	0.51	0.10	0.03	0.73	0.77	0.01
PREADA	**LATBP (Ours)**	0.34	0.29	0.08	0.05	0.10	0.36	0.03
				`Deepseek-R1-Distill-Qwen-7B`				
—	*STD.*	0.38	0.44	0.59	0.56	0.47	0.45	0.15
INADA	AC	0.55	0.63	0.78	0.74	0.66	0.61	0.19
	ESC	0.49	0.56	0.71	0.68	0.58	0.55	0.14
	DSC	0.57	0.65	0.82	0.79	0.69	0.66	0.11
	DeepConf	0.73	0.81	0.97	0.93	0.85	0.88	0.15
PREADA	**LATBP (Ours)**	0.26	0.30	0.38	0.36	0.31	0.29	0.13
				`Qwen3-4B`				
—	*STD.*	0.35	0.41	0.54	0.52	0.45	0.43	0.03
INADA	AC	0.51	0.60	0.73	0.71	0.63	0.59	0.06
	ESC	0.47	0.55	0.68	0.66	0.57	0.54	0.12
	DSC	0.53	0.61	0.75	0.72	0.64	0.60	0.08
	DeepConf	0.69	0.78	0.92	0.89	0.81	0.84	0.12
PREADA	**LATBP (Ours)**	0.24	0.28	0.36	0.34	0.29	0.27	0.11

G.3 Cost Statistics

Table 18 and 19 report the training duration inference latency of predictors in our LATBP. Since they depend solely on the hidden size of the applied language model, we report results on four different models.

Table 18: Training Duration of predictors in our LATBP(**the total of L predictors**).

`Qwen2.5-7B`	`Llama3.1-8B`	`Deepseek-R1-Distill-Qwen-7B`	`Qwen3-4B`
67.4s	79.5s	68.0s	87.4s

Table 19: Inference latency of predictors in our LATBP(**the total of L predictors**).

`Qwen2.5-7B`	`Llama3.1-8B`	`Deepseek-R1-Distill-Qwen-7B`	`Qwen3-4B`
0.023ms	0.029ms	0.025ms	0.031ms

Table 20 reports the average per-sample inference latency of all efficient decoding methods.

Table 20: Average **per-sample** inference latency (seconds) of all methods corresponding to Table **??** and 14 – 16.

Paradigm	Method	MATH500	AMC	AIME24	AIME25	GPQA	MMLU-Pro
				`Qwen2.5-7B`			
—	*STD.*	3.4	5.2	31.2	26.3	9.3	8.5
INADA	AC	5.1	12.1	108.0	74.4	25.1	17.3
	ESC	9.4	9.9	30.0	38.1	21.1	15.1
	DSC	4.9	11.2	57.1	25.5	20.3	31.8
	DeepConf	11.7	13.9	98.6	49.4	31.7	27.3
PREADA	**LATBP (Ours)**	1.6	2.3	25.6	17.9	4.7	3.3
				`Llama3.1-8B`			
—	*STD.*	3.0	4.6	28.3	27.3	7.2	8.9
INADA	AC	5.1	10.4	100.7	95.6	21.5	28.2
	ESC	4.2	9.3	46.4	66.6	17.1	19.2
	DSC	5.2	10.0	46.1	26.8	15.0	25.2
	DeepConf	7.1	12.0	113.8	142.5	30.0	37.0
PREADA	**LATBP (Ours)**	1.2	1.7	12.2	14.2	2.7	3.7
				`Deepseek-R1-Distill-Qwen-7B`			
—	*STD.*	15.7	24.2	204.9	225.4	72.1	64.7
INADA	AC	29.7	67.0	512.2	619.8	188.9	150.1
	ESC	13.2	41.6	448.7	493.6	120.4	124.2
	DSC	33.4	69.9	733.5	450.8	290.6	115.8
	DeepConf	60.6	86.2	1020.4	804.7	274.0	251.0
PREADA	**LATBP (Ours)**	5.8	11.6	145.5	112.7	32.4	31.7
				`Qwen3-4B`			
—	*STD.*	6.5	11.1	65.3	51.4	33.2	30.7
INADA	AC	10.1	29.6	203.7	123.9	70.1	72.5
	ESC	5.7	20.4	126.0	115.6	65.4	68.5
	DSC	8.8	19.0	144.3	142.9	130.1	91.2
	DeepConf	18.0	36.3	302.3	193.3	106.2	153.2
PREADA	**LATBP (Ours)**	2.9	5.8	51.6	28.8	15.3	12.3

H Extended Analysis

H.1 Sensitivity to Sampling Hyperparameters

We further evaluate the sensitivity of LATBP to sampling hyperparameters. We consider four configurations of top-k, top-p, and temperature T:

(a) $(20, 0.95, 0.6)$, our main setup;
(b) $(40, 0.95, 0.6)$;
(c) $(20, 0.8, 0.6)$;
(d) $(20, 0.95, 1.0)$.

We evaluate two settings. The first is *In-Domain (ID) Adaptability*, where the predictor is trained and evaluated under the same sampling configuration. This setting tests whether LATBP remains effective when the sampling hyperparameters are changed consistently across training and decoding. The second is *Out-of-Domain (OOD) Generalization*, where the predictor is trained under one

configuration but evaluated under another. This setting tests whether LATBP remains robust under training–decoding hyperparameter mismatch. Results are reported in Table 21.

In the ID setting, LATBP shows strong adaptability to sampling-hyperparameter changes. Across the four configurations, its accuracy gaps relative to the corresponding *STD.* baselines are $+0.02$, -0.23, -0.24, and -0.07, respectively. All gaps remain within a small range, indicating that LATBP remains stable when k, p, and T are changed consistently between training and decoding.

In the OOD setting, LATBP faces a harder generalization problem, since the predictor is trained and evaluated under different sampling configurations. Nevertheless, the degradation remains mild, with the worst accuracy drop being only 0.50 points. We also observe a rough correlation between cost mismatch and accuracy drop: when the training configuration induces a budget scale farther from the inference-time optimum, performance tends to degrade more. This is expected because the predicted budget is tied to the sampling distribution used to construct training labels. Overall, these results suggest that LATBP is not overly sensitive to sampling hyperparameters and remains robust under moderate training–decoding mismatch.

Table 21: Sensitivity of LATBP to sampling hyperparameters on `Qwen3-4B` with `AIME25`. The upper panel reports *STD.* accuracy under each inference configuration. In the lower panel, **Tr.** and **In.** denote the sampling configurations used for predictor training and decoding, respectively. Gray cells indicate ID evaluation; other cells indicate OOD evaluation.

	Acc. ↑			
STD.	(a) 54.53	(b) 54.18	(c) 54.35	(d) 53.97

Tr. \ In.	$\delta_{\mathcal{M}}$ ↓				$\delta_{\mathcal{T}}$ ↓				*Acc.* ↑			
	(a)	(b)	(c)	(d)	(a)	(b)	(c)	(d)	(a)	(b)	(c)	(d)
(a)	0.17	0.17	0.19	0.18	0.56	0.55	0.59	0.56	54.55	53.81	54.02	53.57
(b)	0.19	0.18	0.19	0.19	0.59	0.60	0.55	0.57	54.34	53.95	54.03	53.93
(c)	0.15	0.15	0.15	0.16	0.54	0.53	0.53	0.57	54.32	53.72	54.11	53.61
(d)	0.21	0.23	0.21	0.23	0.61	0.59	0.61	0.62	54.18	53.74	53.85	53.90

H.2 Ablation Study of LATBP

Hidden Size of MLP Architecture. The only architectural hyperparameter of the predictor is the MLP hidden size, set as $\lfloor rd \rfloor$, where d is the hidden dimension of the language model. In our main experiments, we use $r = 1/8$. We vary r to study how predictor capacity affects LATBP in real-world decoding.

Figures 63 and 64 show the results on `MATH500` and `AIME25`. When $r < 1/8$, decoding accuracy drops to different extents across models, indicating that insufficient predictor capacity harms budget prediction. When $r > 1/8$, performance largely saturates, suggesting that $r = 1/8$ already provides sufficient capacity and further scaling brings limited benefit.

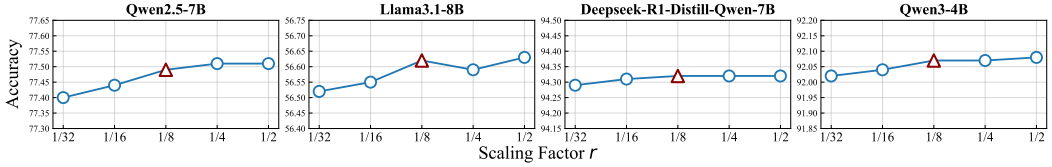


Figure 63: Decoding accuracy under varying scaling factor r of MLP hidden size $\lfloor rd \rfloor$ on the `MATH500` dataset.

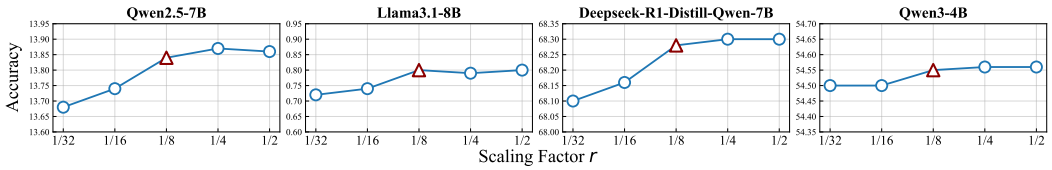


Figure 64: Decoding accuracy under varying scaling factor r of MLP hidden size $\lfloor rd \rfloor$ on the AIME25 dataset.

Training Data Size. We also study the effect of training data size for the predictor. In our main experiments, the predictor is trained with 5k samples. Here, we vary the number of training samples to examine how data scale affects LATBP in real-world decoding.

Figures 65 and 66 report the results on MATH500 and AIME25. Overall, performance improves rapidly with more training data at small scales, but the gains become much slower after 5k samples. At 10k samples, the improvement is marginal, and mild overfitting is even observed for Qwen. These results suggest that 5k samples provide the best efficiency-performance trade-off, and we therefore use 5k as the default training size in our main experiments.

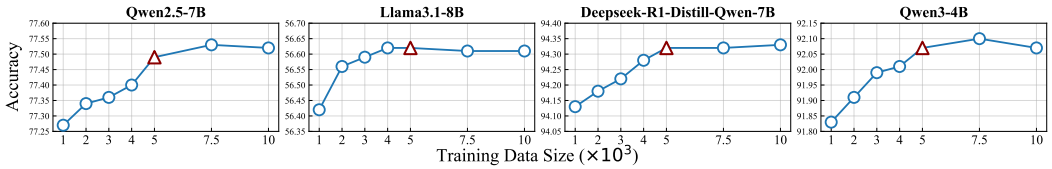


Figure 65: Decoding accuracy under varying training data sizes on the MATH500 dataset.

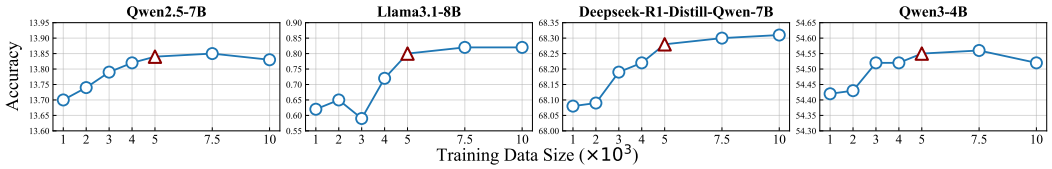


Figure 66: Decoding accuracy under varying training data sizes on the AIME25 dataset.

Inference Weighting Strategy. During inference, LATBP combines the outputs of the L layer-wise predictors into a single budget prediction, as defined in Eq.5. Our theoretical derivation leads to *Inverse Squared Error Weighting* (Eq.7), where more reliable predictors receive larger weights. To verify whether this choice is necessary in practice, we compare it with two simpler alternatives: (1) *Uniform Weighting*, which assigns equal weights to all layers, and (2) *One-hot Weighting*, which uses only the predictor with the lowest validation error.

Table 22 reports the results on all datasets with Qwen3-4B. Inverse Squared Error Weighting consistently achieves the best performance across all datasets, confirming the practical effectiveness of our theoretical weighting rule. Compared with One-hot Weighting, our method benefits from aggregating complementary information across multiple layers rather than relying on a single best layer. Compared with Uniform Weighting, it avoids treating unreliable layers equally, which explains why simple averaging performs the worst. These results suggest that both cross-layer information and reliability-aware weighting are important for stable budget prediction.

Table 22: Comparison of decoding accuracy across inference weighting strategies for LATBP on Qwen3-4B.

	MATH500	AMC	AIME24	AIME25	GPQA	MMLU-Pro	BrowseComp
Inverse Squared Error Weighting (Ours)	92.88	82.18	59.35	54.53	65.24	71.87	11.23
One-hot Weighting	91.54	81.43	56.68	53.92	64.88	71.79	10.23
Uniform Weighting	89.49	80.09	56.63	53.24	64.34	70.82	9.93

H.3 Design Principle of LatBP

Layer-wise Pattern. We train L layer-wise predictors and combine their outputs during inference. A simpler alternative is to concatenate the hidden states from all L layers and train a single predictor. However, this concatenated-state design mixes heterogeneous representations and greatly increases the feature dimension, making the predictor more prone to sparsity and overfitting under limited training data. In contrast, the layer-wise design preserves depth-specific signals by modeling each layer separately.

We compare these two designs to examine whether preserving layer-wise structure is important for LatBP. Table 23 reports the results on all datasets with `Qwen3-4B`. Our layer-wise design consistently achieves stronger decoding accuracy, supporting layer-wise prediction with inference-time combination as a more effective design for LatBP.

Table 23: Decoding accuracy comparison of layer-wise and concatenated-state predictors for LatBP on `Qwen3-4B`.

	MATH500	AMC	AIME24	AIME25	GPQA	MMLU-Pro	BrowseComp
Layer-wise Predictors (ours)	92.88	82.18	59.35	54.53	65.24	71.87	11.23
Single Predictor w/ Concatenated States	90.09	79.80	59.12	52.90	64.23	70.16	10.24

Training Objective. We formulate sample-optimal budget prediction as a regression task. A natural alternative is to treat budgets as discrete labels and train an $N_{\max}$-class classifier. However, budget prediction is inherently distance-sensitive: a small deviation from $N^*_{\boldsymbol{x}}$ usually has limited impact, while a large deviation can substantially affect performance. Regression naturally preserves this distance structure by penalizing larger errors more strongly. In contrast, classification treats budgets as independent classes and does not explicitly model their distances to the target budget.

We compare these two objectives to examine whether preserving distance information is important for LatBP. Table 24 reports the results on `Qwen3-4B`. Regression consistently outperforms classification across all datasets, confirming that distance-aware regression is a more suitable objective for predicting sample-optimal budgets.

Table 24: Comparison of decoding accuracy between regression and classification objectives of predictor training for LatBP on `Qwen3-4B`.

	MATH500	AMC	AIME24	AIME25	GPQA	MMLU-Pro	BrowseComp
Regressor (Ours)	92.88	82.18	59.35	54.53	65.24	71.87	11.23
$N_{\max}$-class Classifier	88.54	79.64	55.28	50.13	62.73	69.22	8.32

I Related Work

I.1 Parallel Thinking

Parallel thinking is a test-time scaling paradigm that mainly consists of two stages: ***multi-path sampling*** and ***answer aggregation*** [25]. In the first stage, stochastic sampling is commonly adopted, where each reasoning path is independently explored from scratch via multinomial sampling [49, 29, 3, 5, 24, 50], including variants that partially relax path independence [39, 9]. In the second stage, majority voting is the most widely used unsupervised aggregation method, offering strong generality, simplicity, and minimal prior assumptions [49, 59, 14, 28]. Beyond majority voting, some methods incorporate additional priors into aggregation, such as confidence or hidden-state signals [24, 12, 53, 52], while others introduce external priors by scoring candidates with reward models [1, 46] or LLMs [23, 59].

Beyond this standard sampling-and-aggregation setting, some studies explore heuristic methods for multi-path exploration, providing finer-grained control during decoding. For example, Beam Search [57] improves lookahead by maintaining multiple hypotheses at each decoding step. Other strategies,

such as Tree-of-Thought (ToT) [63], Skeleton-of-Thought (SoT) [36], and Monte Carlo Tree Search (MCTS) [67, 16, 27, 8], build on stochastic rollouts or structured expansions, introducing additional hyperparameters to control branching, merging, truncation, and back-propagation of sampling paths.

Another line of work investigates training-based methods for further unlocking parallel thinking capabilities in LLMs, including SFT [56, 62], RL [68, 58], and architectural parallelism [6]. Since our focus is on training-free methods, these studies are orthogonal to our work. Overall, parallel thinking provides a promising avenue for unlocking the latent reasoning capabilities of LLMs.

I.2 Efficient Parallel Decoding

Efficient parallel decoding mainly follows two research directions [50]: ***breadth pruning*** and ***depth pruning***. Breadth pruning adaptively reduces the number of parallel samples for each input, and our LATBP belongs to this line. Prior methods, including AC [1], ESC [26], and DSC [48], rely on answer frequency or problem difficulty for dynamic early stopping. However, our experiments show that such decoding-time adaptation can incur substantial latency due to blocked hardware parallelization.

Depth pruning, in contrast, keeps the overall sampling budget fixed while shortening individual reasoning paths. For example, SBoN [44] uses a reward model to score early partial sequences and discard low-scoring paths, ST-BoN [50] truncates unpromising samples based on internal consistency estimates, and Slim-SC [22] prunes highly similar paths on the fly during decoding. These two lines are **orthogonal**: breadth pruning improves efficiency by reducing the number of paths, while depth pruning improves efficiency by reducing the length or cost of each path. **Our LATBP belongs to breadth pruning**, as its efficiency gains come from predicting and allocating sample-specific parallelism levels before decoding.

Although orthogonal, the two directions are not mutually exclusive and can be combined. For example, DeepConf [12] performs path-level truncation within an adaptive early-stopping framework. Similarly, our PREADA paradigm can be seamlessly integrated with depth-pruning methods: once LATBP predicts the optimal parallelism level before decoding, path-level pruning can still be applied during decoding without interfering with the pre-decoding budget allocation.

I.3 Test-time Scaling

Test-time scaling [43] improves reasoning performance by increasing computation at inference time. This paradigm mainly follows two directions [35]: ***sequential scaling*** and ***parallel scaling***. Sequential scaling extends the length of a single chain-of-thought, encouraging slower and more deliberate reasoning. This can elicit cognitive behaviors such as reflection and backtracking [17, 13], thereby increasing the chance of reaching the correct answer. Representative approaches include RL [40, 17, 65], SFT [35, 64, 60], and inference-time prompt forcing [35, 51]. Parallel scaling corresponds to parallel thinking, which is discussed in detail in Appendix I.1. These two paradigms are complementary and together provide a foundation for eliciting stronger reasoning capabilities from LLMs.

I.4 Efficiency Issue in Test-time Scaling

Among TTS paradigms, overthinking [7] is a well-known efficiency issue in sequential scaling. Although the overscaling curse also ultimately results in an efficiency issue, it should not be viewed as a simple parallel counterpart of overthinking [7], even though both belong to the broader TTS paradigm. They are fundamentally different. Overthinking is a ***within-sample*** issue, where excessive computation on a sample creates a trade-off between its own efficacy and efficiency. By contrast, overscaling arises from a bidirectional coupling between the system and samples: individual sample behaviors collectively determine system efficacy, which is then imposed back on the efficiency of all samples. This ***sample–system–sample*** constraint creates cross-sample interactions, making overscaling far more complex.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: In Abstract and Section 1

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: After Section 7

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: In Appendix C.3

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: In Section 2, Section 4.4, and Section 5.2

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: N/A

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: In Section 2, Section 4.4, and Section 5.2

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: In Section G.2

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: In Section 2 and Appendix G.3

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes].

    Justification: After Section 7

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: In Section 2

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.