On the Role of Temperature Sampling in Test-Time Scaling

Yuheng Wu **Thierry Tambe**
Stanford University
{yuhengwu,ttambe}@stanford.edu

Abstract

Large language models (LLMs) can improve reasoning at inference time through test-time scaling (TTS), where multiple reasoning traces are generated and the best one is selected. Prior work shows that increasing the number of samples K steadily improves accuracy. In this paper, we demonstrate that this trend does not hold indefinitely: at large K, further scaling yields no gains, and certain *hard* questions remain unsolved regardless of the number of traces. Interestingly, we find that different sampling temperatures solve different subsets of problems, implying that single-temperature scaling explores only part of a model's potential. We therefore propose scaling along the temperature dimension, which enlarges the reasoning boundary of LLMs. Averaged over Qwen3 (0.6B, 1.7B, 4B, 8B) and five representative reasoning benchmarks (AIME 2024/2025, MATH500, LiveCodeBench, Hi-ToM), temperature scaling yields an additional 7.3 points over single-temperature TTS. Temperature scaling also enables base models to reach performance comparable to reinforcement learning (RL)-trained counterparts, without additional post-training. We further provide a comprehensive analysis of this phenomenon and design a multi-temperature voting method that reduces the overhead of temperature scaling. Overall, our findings suggest that TTS is more powerful than previously thought, and that temperature scaling offers a simple and effective way to unlock the latent potential of base models.

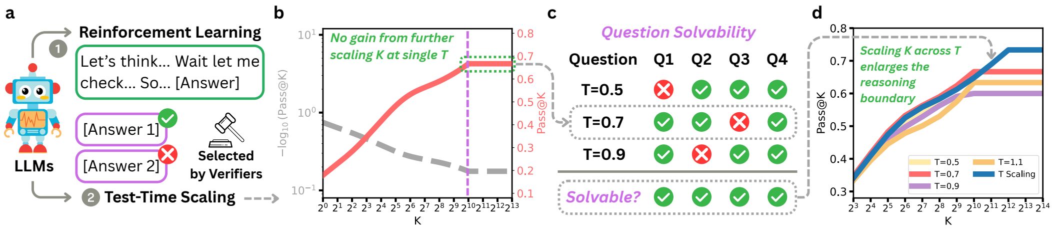
1 Introduction

Large language models (LLMs) have demonstrated strong reasoning capabilities for complex problems at test time [1]. As illustrated in Figure 1a, two main approaches have emerged to achieve such reasoning. The first trains models to produce long reasoning traces with self-reflection and correction, often implemented through reinforcement learning (RL) [2, 3]. While effective, this approach requires costly and time-consuming training [4]. The second, known as test-time scaling (TTS) [5–7], shifts the burden to inference: the model generates multiple reasoning traces in parallel and a verifier selects the most reliable one [8]. Unlike RL, TTS requires no large-scale post-training and generates relatively short, prefix-sharing traces. This enables efficient reuse of the KV cache [9] and offers speed advantages in modern serving systems [10, 11].

Recent studies on TTS have shown that increasing the number of samples K can enhance reasoning performance [12]. As illustrated in Figure 1b, scaling K up to 1,024 shows a clear trend of steadily improving accuracy. However, when we push K further to 13,312, the improvement stops: some questions remain unsolved no matter how many samples are drawn. *If further scaling K brings no improvement, have we reached the ceiling of TTS performance?*

The answer is no. As shown in Figure 1c, we observe an interesting phenomenon: when scaling K up to 1,024 under different sampling temperatures T, the sets of solvable questions differ. For

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: The 1st Workshop on Efficient Reasoning.



Question	Q1	Q2	Q3	Q4
T=0.5	✗	✓	✓	✓
T=0.7	✓	✓	✗	✓
T=0.9	✓	✗	✓	✓
Solvable?	✓	✓	✓	✓

Figure 1: Observations and motivation for temperature scaling in TTS. (a) RL vs. TTS: RL produces long single traces, while TTS generates multiple shorter ones. (b) Pass@K and $-\log(\text{Pass@}K)$ curves at $T = 0.7$ on Qwen3-4B (AIME 2025); no gain beyond $K = 1{,}024$. (c) Question solvability on AIME 2025 for Qwen3-4B: different temperatures solve different subsets of questions. (d) Single-temperature vs. multi-temperature scaling: the latter expands the reasoning boundary.

example, a question unsolved at $T = 0.5$ may become solvable at $T = 0.7$. The model's overall solvable set is therefore larger than the solvable set under any single temperature. This indicates that scaling K at a fixed temperature only explores part of the model's potential. To unlock the full boundary, we scale along the temperature dimension: given any budget, we divide samples evenly across multiple temperatures. As shown in Figure 1d, this multi-temperature strategy achieves a much higher performance upper bound. These results highlight the importance of temperature sampling in TTS.

In this paper, we show that temperature provides a new dimension for scaling at test time. Through extensive experiments across model sizes and datasets, we demonstrate that temperature scaling expands the reasoning boundary of LLMs. This effect arises because, while all temperatures can solve *easy* questions, some *hard* questions are only solvable at specific temperatures. With temperature scaling, a base model can reach performance comparable to RL-trained models. Averaged over Qwen3 (0.6B, 1.7B, 4B, 8B) and five benchmarks (AIME 2024/2025, MATH500, LiveCodeBench, Hi-ToM), temperature scaling yields an additional 7.3 points over single-temperature TTS. We further conduct entropy-based analyses and case studies to understand this phenomenon. Finally, we design a multi-temperature voting method that identifies and exits *easy* questions early, making temperature scaling more efficient. Overall, our contributions are threefold:

- **Scaling test-time compute to a new dimension.** We show that scaling the sampling temperature expands the model's reasoning boundary, revealing a new axis of test-time compute.
- **Analyzing the dynamics of temperature scaling.** Through comprehensive analysis, we show that the enlarged reasoning boundary arises because different temperatures solve different *hard* questions, while *easy* questions can be solved by all temperatures.
- **Designing efficient methods for temperature scaling.** We propose a multi-temperature voting strategy that exits *easy* questions early, reducing overhead while preserving the benefits of temperature scaling across models and datasets.

2 Scaling Temperature at Test Time

In this section, we first introduce the concept of temperature sampling and describe the experimental setup for scaling temperature. We then present the performance improvements achieved through temperature scaling, and compare this approach against further scaling K and RL-based methods.

2.1 Temperature Sampling

Temperature sampling. Temperature-based sampling has long been studied in probabilistic modeling [13], and it remains a core component of decoding in LLMs. At each step of autoregressive generation, the model conditions on the input x and previously generated tokens $y_{1:t-1}$ to yield a distribution over the next token, from which y_t is sampled:

$$y_t \sim p(\cdot \mid x, y_{1:t-1}) = \mathrm{softmax}\left(\frac{f_\theta(x, y_{1:t-1})}{T}\right).$$

Here, $f_\theta(\cdot)$ denotes the model logits and T is the sampling temperature. The temperature rescales the logits before applying the softmax, thereby shaping the probability distribution used for sampling.

Table 1: Results (%) of temperature scaling across models and datasets. Base reports Pass@$1,024$ under $T = 0.6$. $+T$ reports Pass@All when scaling the sampling temperature from 0.0 to 1.2, with 1,024 samples per temperature. Δ denotes the difference between $+T$ and Base.

Models	AIME2025			AIME2024			MATH500			LiveCodeBench			Hi-ToM		
	Base	$+T$	Δ	Base	$+T$	Δ	Base	$+T$	Δ	Base	$+T$	Δ	Base	$+T$	Δ
Qwen3-8B	60.0	66.7	**6.7**	73.3	80.0	**6.7**	97.0	97.8	**0.8**	32.6	40.0	**7.4**	93.0	95.5	**2.5**
Qwen3-4B	60.0	73.3	**13.3**	66.7	76.7	**10.0**	95.5	98.5	**3.0**	36.0	40.0	**4.0**	83.0	92.0	**9.0**
Qwen3-1.7B	46.7	50.0	**3.3**	43.3	50.0	**6.7**	92.5	95.5	**3.0**	29.7	35.4	**5.7**	37.0	55.0	**18.0**
Qwen3-0.6B	20.0	36.7	**16.7**	23.3	30.0	**6.7**	88.1	94.0	**5.9**	25.7	31.4	**5.7**	71.5	82.5	**11.0**

Effect of temperature. The sampling temperature T is non-negative. When $T = 0$, the distribution collapses to a delta on the maximum-logit token, equivalent to deterministic decoding. For $T > 0$, smaller values sharpen the distribution, making generation more deterministic and favoring high-probability tokens, while larger values flatten it, increasing randomness and promoting diversity.

2.2 Experimental Setup

Sampling temperature and number of samples. We vary the sampling temperature from 0.0 to 1.2 in increments of 0.1. At temperature 0.0, we generate a single reasoning trace for each question, whereas at other temperatures we generate 1,024 traces per question.

Datasets and prompts. We evaluate on reasoning benchmarks spanning multiple domains. AIME 2024, AIME 2025, and MATH500 [14] are used to assess mathematical reasoning. LiveCodeBench v6 [15] evaluates code generation, and Hi-ToM [16] focuses on social and logical reasoning. All these benchmarks support automatic output verification. Further details about the datasets and prompts are provided in Appendix A.

Platform and models. All experiments are conducted using vLLM [10] to enable large-batch rollouts. We run our evaluations on a cluster using 64 NVIDIA H100 GPUs. The models include the Qwen3 series (0.6B, 1.7B, 4B, 8B) [17], and Polaris-4B-Preview [18], an RL-trained variant of Qwen3-4B.

Evaluation metrics. Our primary evaluation metric is Pass@K, where K is the number of sampled traces. It measures the probability of obtaining at least one correct answer when sampling K times. Let N be the total number of generated samples and C the number of correct ones, then

$$\text{Pass@}K = 1 - \frac{\binom{N-C}{K}}{\binom{N}{K}}.$$

In addition, the average accuracy Avg@$N = C/N$ approximates the model's correctness probability when N is large. As our aim is to highlight scaling behavior, we use ground-truth verification instead of a reward model (RM) to avoid confounds from RM quality.

AIME 2024/2025 reasoning trace validation. Each AIME problem has an integer answer. It has been argued that LLMs may sometimes arrive at the correct answer by chance rather than through valid reasoning [19]. To address this concern, we conduct additional validation on all problems where a model's Avg@$1,024$ is below 3%. For these problems, the model-generated reasoning traces and human-written reference solutions are jointly reviewed by gpt-5 as an automatic judge. Further details of this validation process are provided in Appendix A.

2.3 Results on Temperature Scaling

Main results. We compare scaling at the default temperature ($T = 0.6$) with multi-temperature scaling. As shown in Table 1, temperature scaling consistently enlarges the reasoning boundary across all models and datasets. Averaged over Qwen3 (0.6B, 1.7B, 4B, 8B) and five benchmarks, it yields an additional 7.3 points over single-temperature TTS. Concretely, on AIME 2025, Qwen3-4B gains 13.3 points after temperature scaling, indicating that any single temperature covers only part of the model's reasoning capability. However, the effect varies by dataset: on MATH500, where Qwen3-8B already performs strongly, the additional gain from temperature scaling is relatively small.

No single temperature works for all questions. Each question shows its own temperature preference. For example, in Figure 2a, one AIME 2025 problem requires $T = 1.1$ with 128 traces to reach

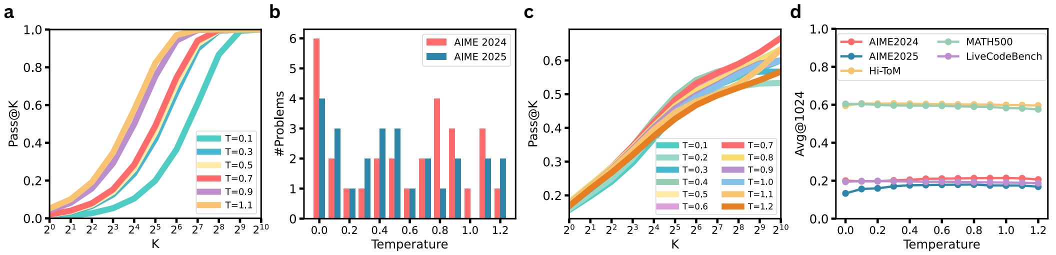


Figure 2: Scaling temperature for test-time compute on Qwen3-4B. (a) Pass@K curves for different temperatures on AIME 2025 Q22. (b) Distribution of preferred temperatures across AIME 2024/2025. (c) Pass@K scaling curves on AIME 2025. (d) Avg@$1,024$ curves across five datasets.

Pass@$K = 1$, while $T = 0.7$ needs 256 traces. Figure 2b further shows the temperature preferences of Qwen3-4B on all AIME 2024/2025 questions: each question has a different optimal temperature, and no single value performs best across the board. Interestingly, while higher temperatures increase diversity and creativity, we find no consistent link to improved reasoning capability.

Pass@K vs. Avg@N. Figure 2c shows Pass@K scaling curves of Qwen3-4B on AIME2025 under different temperatures. When K is small, different temperatures yield similar Pass@K, since models mainly solve *easy* questions that all temperatures can handle. As K grows, however, temperature-specific advantages emerge: a *hard* question favoring one temperature may eventually be solved there but not at others, leading to larger gaps at Pass@$1,024$. In contrast, Figure 2d shows that Avg@$1,024$ remains nearly identical across temperatures, as most *easy* questions are solved by all settings and the ability to solve harder ones is drowned out in this metric.

Remarks. Prior work has reported similar Avg@N curves and argued that, within a range, temperature does not affect LLM reasoning performance [20]. Our results suggest this is not the case in the TTS setting: Avg@N fails to capture the model's ability to solve *hard* questions with low probability, whereas a good verifier can still identify these sparse correct traces [7].

2.4 Scaling Along T vs. Scaling Along K

Understanding dataset difficulty distribution and solvability. Figures 3a and 3b plot the Avg@$1,024$ of each question under two different temperatures, with each point representing one problem. From this view, the questions can be grouped as:

- *Easy*: questions near the upper-right diagonal, solved by both temperatures with high probability.
- *Medium*: points near the lower-left diagonal, harder but still solved by both temperatures.
- *Hard*: points lying on the axes, solvable at one temperature with low probability but not the other.
- *Impossible*: points at the origin, unsolved regardless of temperature.

We observe that many questions cluster along the upper-right diagonal: these are simple problems that any temperature can solve, explaining why Avg@N shows little difference across temperatures. Some questions remain at the origin: these are beyond the reach of temperature scaling and likely require training rather than TTS.

Scaling across temperatures unlocks latent capability. The axis points (*hard* problems) illustrate why temperature scaling is effective: sampling across multiple temperatures ensures that these problems enter the solvable set, whereas sampling under a single temperature collapses all solvable *hard* problems from other temperatures back to the origin.

Further scaling the number of samples does not help. As shown in Figure 1b, increasing K from 1,024 to 13,312 does not solve any additional problems. This suggests that the TTS curve has two phases: an initial regime where more samples improve performance, followed by a plateau where no further gains are observed. In contrast, Figure 3c shows that scaling temperature yields a 6.67% improvement, demonstrating that temperature provides a meaningful additional dimension for TTS.

Remarks. When scaling K to 13,312, we initially observe improvements, with the model producing correct final answers for questions unsolved under a smaller budget. However, after gpt-5-based trace verification, these cases are found to be invalid: the model is merely more likely to guess the correct

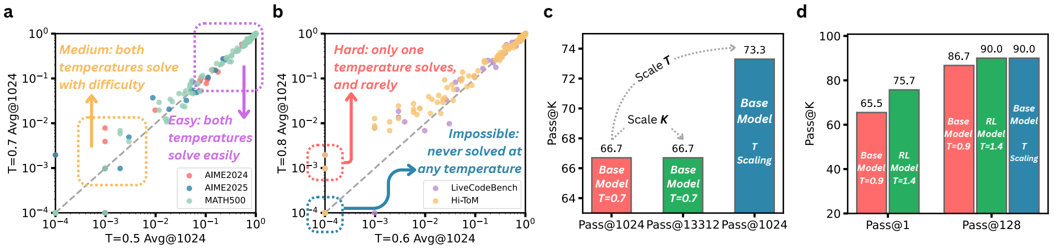


Figure 3: Comparison of scaling along K and scaling along T. (a) Correlation of Avg@$1,024$ across two temperatures on AIME2025, Qwen3-4B. (b) Correlation of Avg@$1,024$ across two temperatures on AIME2025, Qwen3-8B. (c) Scaling K vs. scaling T on AIME2025, Qwen3-4B. (d) Temperature scaling vs. RL-trained model on AIME2025, Qwen3-4B, thinking mode.

answer with more samples. This highlights that future work on RL and TTS should account not only for final-answer accuracy but also for the validity of reasoning traces.

2.5 Comparison with RL-Trained Methods

Setup. We evaluate the RL-trained Polaris-4B-Preview model against its base model, Qwen3-4B. Following the setup suggested by [18], we run the RL model on AIME 2025 with temperature $T = 1.4$, a maximum context length of 96K tokens, and the "thinking" mode enabled. To ensure fairness, we apply the same configuration to the base model: scaling temperature from 0.0 to 1.4, enabling "thinking" mode, and setting $N = 128$ samples per question.

Scaling K cannot make the base model comparable to RL. As shown in Figure 3d, the RL-trained model consistently outperforms the base model when scaling the number of samples, with a clear advantage under small budgets (e.g., Pass@1). Although the performance gap narrows as K increases, the base model never fully catches up.

Further scaling T can make the base model comparable to RL. As shown in Figure 3d, scaling across temperatures raises the base model to Pass@All performance on par with the RL-trained model. The overall success rates are similar, though they differ in which questions remain unsolved: the base model fails on Q14, Q15, and Q28, while the RL model fails on Q13, Q14, and Q15. Thus, temperature scaling extends the base model's reasoning boundary to match RL performance.

Remarks. There is an active debate on whether RL brings genuinely new capabilities to LLMs [4]. Some prior work argues that scaling K sufficiently can erase or even reverse the advantage of RL [21], while others contend that many of the base model's large-K successes are merely guesses, and after verifier filtering the RL-trained model remains stronger [19]. Our findings align with the latter: scaling K narrows the gap but does not eliminate it. However, when we further scale across T, the base model becomes comparable to RL. In this sense, exploring both K and T dimensions reveals a broader reasoning boundary than either alone.

3 Deep Analysis of Temperature Scaling

In this section, we first present the entropy dynamics underlying temperature scaling, then provide a case study to illustrate these behaviors in practice, and finally discuss the strengths and limitations of temperature scaling.

3.1 Learning the Entropy Dynamics of Temperature Scaling

Entropy. We measure model uncertainty using the entropy of the next-token distribution. At each step, given logits $f_\theta(x, y_{1:t-1})$, we compute

$$H = -\sum_{y} p(y \mid x, y_{1:t-1}) \log p(y \mid x, y_{1:t-1}), \quad p(y \mid x, y_{1:t-1}) = \mathrm{softmax}(f_\theta(x, y_{1:t-1})).$$

We report the average entropy H across all generated tokens. Low H indicates sharp logits and high confidence, while high H reflects uncertainty. Here, it is always computed from the model's

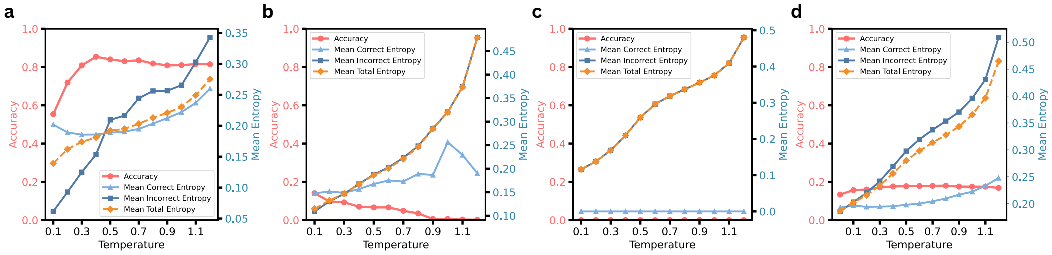


Figure 4: Entropy dynamics of Qwen3-4B across temperatures on AIME 2025. (a) Q16. (b) Q29. (c) Q11. (d) Across the whole dataset.

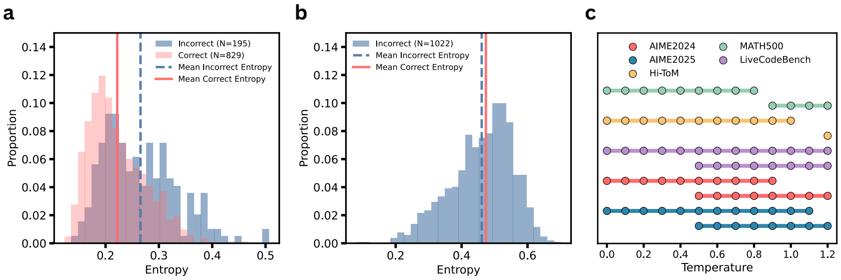


Figure 5: Entropy distributions and temperature subset. (a) An *easy* problem (AIME 2025 Q16, Qwen3-4B). (b) A *hard* problem (AIME 2025 Q25, Qwen3-4B). (c) Temperature minimal subset for Qwen3-4B.

untempered softmax. As shown in Figure 4a–c, average H rises with T: higher T increases the chance of selecting low-probability tokens, raising the sequence-level H.

Entropy dynamics of correct vs. incorrect traces. We analyze the average entropy of all traces, as well as the correct and incorrect subsets. For *easy/medium* questions (Figure 4a), correct-trace entropy increases smoothly with temperature, while incorrect-trace entropy rises much faster. At low T, correct traces show higher entropy, reflecting greater diversity; at high T, they show lower entropy, reflecting coherence when the model answers correctly. For *impossible* questions (Figure 4c), in contrast, the entropy of all traces increases rapidly as T grows. At the dataset level (Figure 4d), correct traces consistently maintain lower entropy than incorrect ones at higher temperatures, suggesting that *the model **seems** to "know" when it is answering correctly.* But is this always the case?

When does the model "know it knows"? The entropy gap between correct and incorrect traces holds reliably only for *easy/medium* questions. As shown in Figure 5a, for an *easy* problem at $T = 1.0$, the entropy distribution of correct traces is clearly shifted to the left compared to incorrect ones. However, for a *hard* problem (Figure 5b), where only 2 out of 1,024 traces are correct, the entropy of correct traces is not always lower than that of incorrect ones. The model "knows it knows" only when the question is *easy*, but not when it is *hard*. This explains why in Figure 4d, where *easy* problems dominate, the aggregate effect makes it appear that correct traces always have lower entropy, drowning out the behavior on *hard* problems.

Remarks. Recent work has explored uncertainty-guided decoding [22, 23], where high-entropy traces are discarded under the assumption they are unlikely correct [24]. Our findings suggest this assumption holds for *easy/medium* questions, where correct traces typically align with lower entropy. However, for *hard* questions, correct traces do not necessarily exhibit low entropy, indicating uncertainty alone is insufficient as a universal signal.

3.2 Case Study

Problem. We illustrate the effect of sampling temperature using AIME 2025 Q24. The task is to determine the number of zeros of the function $f(x) = \sin\big(7\pi \sin(5x)\big), 0 < x < 2\pi$. This problem essentially has a single correct reasoning path: one must reduce the condition $f(x) = 0$ to $\sin(5x) = k/7$ with $k \in \{-7, \ldots, 7\}$, and count the corresponding solutions over five periods.

Case study. At temperature $T = 0.7$, the model solved the problem in 2 out of 1,024 samples. The successful traces explicitly reduced the condition to $\sin(5x) = k/7$, enumerated the valid k values, and counted their solutions within $(0, 2\pi)$. In contrast, at $T = 0.9$, none of the sampled traces followed this reasoning structure; typical attempts instead guessed the number of zeros by listing approximate intersection points, without connecting them to the $k/7$ values, and thus failed to reach the correct count. This illustrates that a *hard* question may be solvable under one temperature but not another, highlighting the need for temperature scaling.

3.3 Strengths and Limitations of Temperature Scaling

Strengths. As shown in Figure 1d, the most notable strength of temperature scaling is that it achieves a higher upper bound when K is large, revealing a larger reasoning boundary. A second advantage is that distributing samples across temperatures does not substantially deviate from any single-temperature Pass@K curve when K is small. This is because improvements at small budgets mainly come from solving *easy* questions, for which accuracy is largely unaffected by the choice of temperature (as also shown in Figures 2d and 3a,b).

Limitations. The main limitation of temperature scaling lies in computational efficiency. For example, using 12 temperatures requires roughly 12× the compute of single-temperature scaling to reach the expanded reasoning boundary. This overhead is less of a concern when a strong verifier is available: most *easy* questions can be solved with little computation and verified early, allowing the budget to be focused on *hard* ones. However, such verifiers are not always accessible. In the next section, we investigate whether some of this cost can be saved even *without* verifiers.

4 Design Test-Time Methods with Temperature Scaling

In this section, we first analyze whether all temperatures and all questions are necessary for temperature scaling. We then design an algorithm for more efficient temperature sampling.

4.1 Are All Temperatures and Questions Necessary for Scaling?

Finding a minimal subset of temperatures. A natural question is whether we must sample from all available temperatures to achieve the expanded upper bound, or whether a smaller subset is sufficient. As shown in Figure 5c, we evaluate subsets formed by gradually adding temperatures from either the low or high end. The results show that starting from higher temperatures requires a smaller subset to reach the upper bound; traces generated at low temperatures can also be obtained at higher ones. Based on this, we select a subset that generalizes across models and datasets, excluding very low temperatures (0.1–0.3).

Early exit for *easy* questions under temperature scaling. Figures 3a and 3b show that *easy* questions require far fewer samples to solve and can be answered reliably at any temperature with high probability. This observation motivates a simple strategy: avoid redundant sampling on such problems by using a voting-based mechanism across temperatures. In the following, we describe how multi-temperature voting can serve as an early-exit method.

4.2 Algorithm for Efficient Temperature Scaling

Method overview. As illustrated in Figure 6, for each question we maintain per-temperature candidate answer pools. In each round, every temperature contributes one new trace per active question, and the corresponding answers are recorded. We first perform *intra-temperature* voting: for each temperature, the most frequent answer is selected, and if its vote count does not reach the intra-threshold τ_{intra}, that temperature is deemed not yet confident and sampling continues. Only when all temperatures satisfy the intra-threshold do we proceed to *cross-temperature* voting, where the majority answers from each temperature are voted across temperatures. If the winning answer reaches the cross-threshold τ_{cross}, the question is marked as *easy* and exits early. Otherwise, sampling continues in the next round. The complete algorithm is provided in Appendix C.

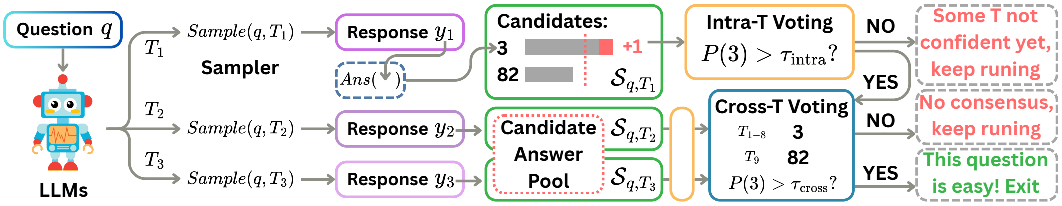


Figure 6: Overview of the voting algorithm for efficient temperature scaling. Each temperature maintains its own candidate pool and first performs intra-temperature voting with threshold $\tau_{\text{intra}} = 0.8$. Only if all temperatures pass this stage do we proceed to cross-temperature voting, where majority votes are aggregated across temperatures with threshold $\tau_{\text{cross}} = 1.0$. Questions that meet both criteria are marked as *easy* and exit early.

Table 2: Results across five datasets. Base reports Pass@$1,024$ at $T = 0.6$. $+T$ reports results with full temperature scaling. $+Ours$ denotes our efficient temperature-scaling method. ΔC shows the reduction in computation cost relative to full temperature scaling.

Models	AIME2025				AIME2024				MATH500			
	Base	$+T$	$+Ours$	ΔC	Base	$+T$	$+Ours$	ΔC	Base	$+T$	$+Ours$	ΔC
Qwen3-8B	60.0	66.7	66.7	**-31.6%**	73.3	80.0	80.0	**-31.6%**	97.0	97.8	97.8	**-54.4%**
Qwen3-4B	60.0	73.3	73.3	**-33.5%**	66.7	76.7	76.7	**-31.6%**	95.5	98.5	98.5	**-49.5%**
Qwen3-1.7B	46.7	50.0	50.0	**-27.2%**	43.3	50.0	50.0	**-27.2%**	92.5	95.5	95.5	**-36.3%**
Qwen3-0.6B	20.0	36.7	36.7	**-25.0%**	23.3	30.0	30.0	**-25.0%**	88.1	94.0	94.0	**-26.0%**

Models	LiveCodeBench				Hi-ToM				Average			
	Base	$+T$	$+Ours$	ΔC	Base	$+T$	$+Ours$	ΔC	Base	$+T$	$+Ours$	ΔC
Qwen3-8B	32.6	40.0	40.0	**-35.0%**	93.0	95.5	94.5	**-78.7%**	71.2	76.0	75.8	**-46.3%**
Qwen3-4B	36.0	40.0	40.0	**-32.8%**	83.0	92.0	86.5	**-32.8%**	68.2	76.1	75.0	**-36.0%**
Qwen3-1.7B	29.7	35.4	35.4	**-29.9%**	37.0	55.0	42.0	**-78.3%**	49.8	57.2	54.6	**-39.8%**
Qwen3-0.6B	25.7	31.4	31.4	**-26.1%**	71.5	82.5	81.5	**-32.4%**	45.7	54.9	54.7	**-26.9%**

Remarks. It is well known that the answer most favored by the model is not always correct [5]. We acknowledge this limitation. However, our goal here is not to aggregate answers for final prediction, but to identify which questions are *easy* and can exit early. Moreover, the use of multi-temperature voting helps smooth out spurious signals, and the relatively high thresholds we adopt ($\tau_{\text{intra}} = 0.8$, $\tau_{\text{cross}} = 1.0$) provide additional robustness against noise.

5 Experiments

In this section, we evaluate the proposed method for more efficient temperature sampling. We then analyze the results and discuss directions for future work.

5.1 Results and Analysis

Main results. As shown in Table 2, our method reduces the computation required for temperature scaling across tasks, while maintaining nearly the same performance. The efficiency gains come from excluding very low temperatures and enabling early exit on *easy* questions. For example, on MATH500, Qwen3-8B achieves a 54.4% reduction in computation with negligible loss in Pass@All. Notably, Hi-ToM shows a different pattern: as a belief-allocation reasoning task, it can be solved by powerful models but also occasionally by weaker ones for spurious reasons, leading to less consistent scaling behavior.

A powerful model is what we need. As shown in Table 2, the overall computation savings also follow a scaling trend. A strong yet compact model, such as Qwen3-8B, is particularly well-suited for temperature scaling: it classifies more questions as *easy*, enabling our method to evict them early for efficiency and apply temperature scaling only to the *hard* ones. By contrast, models that are too small struggle even on *easy* questions, leaving little room for efficient scaling.

5.2 Discussion

Alternative strategies for evicting *easy* questions. Beyond multi-temperature voting, we also explored using entropy signals to identify questions that can be exited early. If the entropy remains very low and changes little across temperatures, the question is likely *easy*; if the entropy is very high, the question is likely *impossible*, and repeated sampling will not help. However, we find that entropy distributions vary significantly across datasets and domains, making it difficult to design a universal entropy-based eviction strategy.

Temperature sampling in feedback-based workflows. In this paper, we focus on internal model signals to evict *easy* questions. In real applications, this is less of a concern: one can simply call a powerful verifier after each run, and if a trace is deemed correct, the question can be evicted immediately. This would allow temperature sampling to focus only on unsolved questions, yielding substantial savings. However, in such workflows the verification cost itself becomes a critical factor. Regardless, we believe temperature sampling remains a simple yet powerful way to explore the reasoning boundary of base models. An interesting direction for future work is to study how variable temperature within a single trace may further influence reasoning ability.

6 Related Work

TTS for LLMs. Multi-trace TTS [5, 6, 12, 7, 25, 26] generates multiple candidate completions in parallel and selects the best one using either a verifier [27–33, 8] or voting-based approaches [34, 35]. This approach can be further combined with search algorithms [36–41], which interleave generation and selection in a step-by-step manner to progressively improve the final output.

Recent studies have also explored the connection between RL and TTS [42, 43, 21, 19, 4]. In parallel, other works investigate how to teach models to dynamically choose between single-trace and multi-trace reasoning strategies [44–47]. Recent work also incorporates TTS into reasoning workflows [48–50].

Sampling methods for TTS. A typical sampling pipeline consists of three components: the prompt input to the model, the stochastic sampling procedure, and post-processing of the logits. Prior work [51, 52] has shown that the design of the prompt can influence the inference performance. Based on temperature sampling [13], many works have explored new strategies for sampling [53–59] and logit truncation [60–64] to better balance diversity and quality. Recently, **(author?)** [20] reported that temperature has limited impact on model reasoning capability. Our results suggest this is not the case when using Pass@K as the metric. Meanwhile, **(author?)** [65] found that the optimal temperature varies across datasets, a pattern we also observe in Figure 2d.

More related work on single-trace TTS, efficient algorithms, and serving systems can be found in Appendix D.

7 Conclusion

We revisit TTS for reasoning in LLMs and show that scaling the number of samples K alone has diminishing returns: once K is large, accuracy plateaus and some questions remain unsolved. In contrast, scaling the sampling temperature expands the reasoning boundary, as different temperatures unlock different *hard* problems while *easy* ones are solved universally. This makes temperature scaling a simple yet powerful complement to traditional TTS, enabling base models to match RL-trained counterparts without additional training. Through entropy-based analysis and case studies, we further characterize the dynamics behind this effect. Finally, we propose efficient multi-temperature voting methods that cut the overhead of temperature scaling by exiting early on *easy* questions. Overall, our results highlight temperature scaling as an effective and practical tool to unlock the latent reasoning capabilities of LLMs.

References

[1] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[2] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi *et al.*, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," arXiv.org, 2025.

[3] Z. Yang, X. Luo, Z. Wang, D. Han, Z. He, D. Li, and Y. Xu, "Do not let low-probability tokens over-dominate in rl for llms," arXiv.org, 2025.

[4] M. Liu, S. Diao, X. Lu, J. Hu, X. Dong, Y. Choi, J. Kautz, and Y. Dong, "Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models," arXiv.org, 2025.

[5] B. Brown, J. Juravsky, R. Ehrlich, R. Clark, Q. V. Le, C. Ré, and A. Mirhoseini, "Large language monkeys: Scaling inference compute with repeated sampling," arXiv.org, 2024.

[6] C. V. Snell, J. Lee, K. Xu, and A. Kumar, "Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning," in *The Thirteenth International Conference on Learning Representations*, 2025.

[7] E. Zhao, P. Awasthi, and S. Gollapudi, "Sample, scrutinize and scale: Effective inference-time search by scaling verification," in *Forty-second International Conference on Machine Learning*, 2025.

[8] J. Saad-Falcon, B. E. Kelly, M. F. Chen, T.-H. Huang, B. McLaughlin, T. Bhathal, S. Zhu, B. Athiwaratkun, F. Sala, S. Linderman, A. Mirhoseini, and C. Ré, "Shrinking the generation-verification gap with weak verifiers," arXiv.org, 2025.

[9] C. Hooper, S. Kim, S. Moon, K. Dilmen, M. Maheswaran, N. Lee, M. W. Mahoney, S. Shao, K. Keutzer, and A. Gholami, "Ets: Efficient tree search for inference-time scaling," arXiv.org, 2025.

[10] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," in *Proceedings of the 29th symposium on operating systems principles*, 2023, pp. 611–626.

[11] L. Zheng, L. Yin, Z. Xie, C. L. Sun, J. Huang, C. H. Yu, S. Cao, C. Kozyrakis, I. Stoica, J. E. Gonzalez *et al.*, "Sglang: Efficient execution of structured language model programs," *Advances in neural information processing systems*, vol. 37, pp. 62 557–62 583, 2024.

[12] R. Schaeffer, J. Kazdan, J. Hughes, J. Juravsky, S. Price, A. Lynch, E. Jones, R. Kirk, A. Mirhoseini, and S. Koyejo, "How do large language monkeys get their power (laws)?" in *Forty-second International Conference on Machine Learning*, 2025.

[13] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for boltzmann machines," *Cognitive Science*, vol. 9, no. 1, pp. 147–169, 1985.

[14] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt, "Measuring mathematical problem solving with the math dataset," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[15] N. Jain, K. Han, A. Gu, W.-D. Li, F. Yan, T. Zhang, S. Wang, A. Solar-Lezama, K. Sen, and I. Stoica, "Livecodebench: Holistic and contamination free evaluation of large language models for code," in *The Thirteenth International Conference on Learning Representations*, 2025.

[16] Y. Wu, Y. He, Y. Jia, R. Mihalcea, Y. Chen, and N. Deng, "Hi-ToM: A benchmark for evaluating higher-order theory of mind reasoning in large language models," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, Dec. 2023.

[17] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv *et al.*, "Qwen3 technical report," arXiv.org, 2025.

[18] C. An, Z. Xie, X. Li, L. Li, J. Zhang, S. Gong, M. Zhong, J. Xu, X. Qiu, M. Wang, and L. Kong, "Polaris: A post-training recipe for scaling reinforcement learning on advanced reasoning models," 2025.

[19] X. Wen, Z. Liu, S. Zheng, Z. Xu, S. Ye, Z. Wu, X. Liang, Y. Wang, J. Li, Z. Miao, J. Bian, and M. Yang, "Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms," arXiv.org, 2025.

[20] M. Renze, "The effect of sampling temperature on problem solving in large language models," in *Findings of the association for computational linguistics: EMNLP 2024*, 2024, pp. 7346–7356.

[21] Y. Yue, Z. Chen, R. Lu, A. Zhao, Z. Wang, Y. Yue, S. Song, and G. Huang, "Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model?" arXiv.org, 2025.

[22] S. Kadavath, T. Conerly, A. Askell, T. Henighan, D. Drain, E. Perez, N. Schiefer, Z. Hatfield-Dodds, N. DasSarma, E. Tran-Johnson *et al.*, "Language models (mostly) know what they know," arXiv.org, 2022.

[23] Z. Kang, X. Zhao, and D. Song, "Scalable best-of-n selection for large language models via self-certainty," arXiv.org, 2025.

[24] Y. Fu, X. Wang, Y. Tian, and J. Zhao, "Deep think with confidence," arXiv.org, 2025.

[25] R. Liu, J. Gao, J. Zhao, K. Zhang, X. Li, B. Qi, W. Ouyang, and B. Zhou, "Can 1b llm surpass 405b llm? rethinking compute-optimal test-time scaling," in *Workshop on Reasoning and Planning for Large Language Models at the Thirteenth International Conference on Learning Representations*, 2025.

[26] A. Khairi, D. D'souza, Y. Shen, J. Kreutzer, and S. Hooker, "When life gives you samples: The benefits of scaling up inference compute for multilingual llms," arXiv.org, 2025.

[27] P. Wang, L. Li, Z. Shao, R. Xu, D. Dai, Y. Li, D. Chen, Y. Wu, and Z. Sui, "Math-shepherd: Verify and reinforce llms step-by-step without human annotations," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 9426–9439.

[28] Z. Gou, Z. Shao, Y. Gong, Y. Yang, N. Duan, W. Chen *et al.*, "Critic: Large language models can self-correct with tool-interactive critiquing," in *The Twelfth International Conference on Learning Representations*, 2024.

[29] Z. Sun, L. Yu, Y. Shen, W. Liu, Y. Yang, S. Welleck, and C. Gan, "Easy-to-hard generalization: Scalable alignment beyond human supervision," *Advances in Neural Information Processing Systems*, vol. 37, pp. 51 118–51 168, 2024.

[30] L. Sun, C. Liu, X. Ma, T. Yang, W. Lu, and N. Wu, "Freeprm: Training process reward models without ground truth process labels," arXiv.org, 2025.

[31] A. Singh, K. Arora, S. Keh, J. Mercat, T. Hashimoto, C. Finn, and A. Kumar, "Improving test-time search for llms with backtracking against in-context value verifiers," in *Workshop on Reasoning and Planning for Large Language Models at the Thirteenth International Conference on Learning Representations*, 2025.

[32] J. Guo, Z. Chi, L. Dong, Q. Dong, X. Wu, S. Huang, and F. Wei, "Reward reasoning model," arXiv.org, 2025.

[33] F. E. Dorner, Y. Chen, A. F. Cruz, and F. Yang, "Roc-n-reroll: How verifier imperfection affects test-time scaling," arXiv.org, 2025.

[34] X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," in *The Eleventh International Conference on Learning Representations*, 2023.

[35] W. Wang, Y. Wang, and H. Huang, "Ranked voting based self-consistency of large language models," arXiv.org, 2025.

[36] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, “Tree of thoughts: Deliberate problem solving with large language models,” *Advances in neural information processing systems*, vol. 36, pp. 11 809–11 822, 2023.

[37] Z. Bi, K. Han, C. Liu, Y. Tang, and Y. Wang, “Forest-of-thought: Scaling test-time compute for enhancing llm reasoning,” in *Forty-second International Conference on Machine Learning*, 2025.

[38] G. Chatziveroglou, “A*-decoding: Token-efficient inference scaling,” arXiv.org, 2025.

[39] J. Qiu, Y. Lu, Y. Zeng, J. Guo, J. Geng, H. Wang, K. Huang, Y. Wu, and M. Wang, “Treebon: Enhancing inference-time alignment with speculative tree-search and best-of-n sampling,” arXiv.org, 2024.

[40] D. Zhang, S. Zhoubian, Z. Hu, Y. Yue, Y. Dong, and J. Tang, “Rest-mcts*: Llm self-training via process reward guided tree search,” in *Proceedings of the 38th International Conference on Neural Information Processing Systems*, 2024, pp. 64 735–64 772.

[41] Q. Lin, B. Xu, Z. Li, Z. Hao, K. Zhang, and R. Cai, “Leveraging constrained monte carlo tree search to generate reliable long chain-of-thought for mathematical reasoning,” arXiv.org, 2025.

[42] K. Sareen, M. M. Moss, A. Sordoni, R. Agarwal, and A. Hosseini, “Putting the value back in rl: Better test-time scaling by unifying llm reasoners with verifiers,” arXiv.org, 2025.

[43] Y. Zuo, K. Zhang, L. Sheng, S. Qu, G. Cui, X. Zhu, H. Li, Y. Zhang, X. Long, E. Hua, B. Qi, Y. Sun, Z. Ma, L. Yuan, N. Ding, and B. Zhou, “Ttrl: Test-time reinforcement learning,” arXiv.org, 2025.

[44] X. Yang, Y. An, H. Liu, T. Chen, and B. Chen, “Multiverse: Your language models secretly decide how to parallelize and merge generation,” arXiv.org, 2025.

[45] E. Biju, S. Talaei, Z. Huang, M. Pourreza, A. Mirhoseini, and A. Saberi, “Sprint: Enabling interleaved planning and parallelized execution in reasoning models,” arXiv.org, 2025.

[46] T. Jin, E. Y. Cheng, Z. Ankner, N. Saunshi, B. M. Elias, A. Yazdanbakhsh, J. Ragan-Kelley, S. Subramanian, and M. Carbin, “Learning to keep a promise: Scaling language model decoding parallelism with learned asynchronous decoding,” in *Forty-second International Conference on Machine Learning*, 2025.

[47] Y. Wang, L. Zhang, Y. Bai, M. T. Chiu, Z. Hu, M. Zhang, Q. Dong, Y. Yin, S. Amirghodsi, and Y. Fu, “Cautious next token prediction,” in *Findings of the Association for Computational Linguistics: ACL 2025*, W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, Eds., Jul. 2025.

[48] J. Saad-Falcon, A. G. Lafuente, S. Natarajan, N. Maru, H. Todorov, E. K. Guha, E. K. Buchanan, M. F. Chen, N. Guha, C. Re *et al.*, “An architecture search framework for inference-time techniques,” in *Forty-second International Conference on Machine Learning*, 2025.

[49] S. Chakraborty, M. Pourreza, R. Sun, Y. Song, N. Scherrer, F. Huang, A. S. Bedi, A. Beirami, J. Gu, H. Palangi, and T. Pfister, “On the role of feedback in test-time scaling of agentic ai workflows,” arXiv.org, 2025.

[50] J. Shen, H. Bai, L. Zhang, Y. Zhou, A. Setlur, S. Tong, D. Caples, N. Jiang, T. Zhang, A. Talwalkar, and A. Kumar, “Thinking vs. doing: Agents that reason by scaling test-time interaction,” arXiv.org, 2025.

[51] T. Wang, Z. Liu, Y. Chen, J. Light, H. Chen, X. Zhang, and W. Cheng, “Diversified sampling improves scaling llm inference,” arXiv.org, 2025.

[52] Y. Liu, Z. Li, Z. Fang, N. Xu, R. He, and T. Tan, “Rethinking the role of prompting strategies in llm test-time scaling: A perspective of probability theory,” arXiv.org, 2025.

[53] C. Meister, T. Pimentel, G. Wiher, and R. Cotterell, “Locally typical sampling,” *Transactions of the Association for Computational Linguistics*, vol. 11, 2023.

[54] Y. Li, F. Mi, Y. Li, Y. Wang, B. Sun, S. Feng, and K. Li, "Dynamic stochastic decoding strategy for open-domain dialogue generation," in *Findings of the Association for Computational Linguistics: ACL 2024*, L.-W. Ku, A. Martins, and V. Srikumar, Eds., Aug. 2024.

[55] Y. Zhu, J. Li, G. Li, Y. Zhao, J. Li, Z. Jin, and H. Mei, "Hot or cold? adaptive temperature sampling for code generation with large language models," in *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, 2024, pp. 437–445.

[56] S. Zhang, Y. Bao, and S. Huang, "Edt: Improving large language models' generation by entropy-based dynamic temperature sampling," arXiv.org, 2024.

[57] C. Cai, X. Zhao, Y. Du, H. Liu, and L. Li, "t^2 of thoughts: Temperature tree elicits reasoning in large language models," arXiv.org, 2024.

[58] S. Dhuliawala, I. Kulikov, P. Yu, A. Celikyilmaz, J. Weston, S. Sukhbaatar, and J. Lanchantin, "Adaptive decoding via latent preference optimization," arXiv.org, 2024.

[59] Y. Li, J. Zhang, S. Feng, P. Yuan, X. Wang, J. Shi, Y. Zhang, C. Tan, B. Pan, Y. Hu, and K. Li, "Revisiting self-consistency from dynamic distributional alignment perspective on answer aggregation," in *Findings of the Association for Computational Linguistics: ACL 2025*, Jul. 2025.

[60] A. Fan, M. Lewis, and Y. Dauphin, "Hierarchical neural story generation," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, I. Gurevych and Y. Miyao, Eds., Jul. 2018.

[61] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," in *The Eighth International Conference on Learning Representations*, 2020.

[62] S. Basu, G. S. Ramachandran, N. S. Keskar, and L. R. Varshney, "Mirostat: A neural text decoding algorithm that directly controls perplexity," in *The ninth International Conference on Learning Representations*, 2021.

[63] J. Hewitt, C. Manning, and P. Liang, "Truncation sampling as language model desmoothing," in *Findings of the Association for Computational Linguistics: EMNLP 2022*, Y. Goldberg, Z. Kozareva, and Y. Zhang, Eds., Dec. 2022.

[64] N. N. Minh, A. Baker, C. Neo, A. G. Roush, A. Kirsch, and R. Shwartz-Ziv, "Turning up the heat: Min-p sampling for creative and coherent llm outputs," in *The Thirteenth International Conference on Learning Representations*, 2025.

[65] W. Du, Y. Yang, and S. Welleck, "Optimizing temperature for language models with multi-sample inference," in *Forty-second International Conference on Machine Learning*, 2025.

[66] M. Sclar, J. Dwivedi-Yu, M. Fazel-Zarandi, Y. Tsvetkov, Y. Bisk, Y. Choi, and A. Celikyilmaz, "Explore theory of mind: program-guided adversarial data generation for theory of mind reasoning," in *The Thirteenth International Conference on Learning Representations*, 2025.

[67] Y. Wu, W. Guo, Z. Liu, H. Ji, Z. Xu, and D. Zhang, "How large language models encode theory-of-mind: a study on sparse parameter patterns," *npj Artificial Intelligence*, vol. 1, no. 1, p. 20, 2025.

[68] Y. Wu, J. Xie, D. Zhang, and Z. Xu, "Del-tom: Inference-time scaling for theory-of-mind reasoning via dynamic epistemic logic," arXiv.org, 2025.

[69] N. Muennighoff, Z. Yang, W. Shi, X. L. Li, L. Fei-Fei, H. Hajishirzi, L. Zettlemoyer, P. Liang, E. Candès, and T. Hashimoto, "s1: Simple test-time scaling," arXiv.org, 2025.

[70] S. Wang, L. Yu, C. Gao, C. Zheng, S. Liu, R. Lu, K. Dang, X. Chen, J. Yang, Z. Zhang *et al.*, "Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning," arXiv.org, 2025.

[71] D. Cheng, S. Huang, X. Zhu, B. Dai, W. X. Zhao, Z. Zhang, and F. Wei, “Reasoning with exploration: An entropy perspective,” arXiv.org, 2025.

[72] D. Li, S. Cao, T. Griggs, S. Liu, X. Mo, E. Tang, S. Hegde, K. Hakhamaneshi, S. G. Patil, M. Zaharia, J. E. Gonzalez, and I. Stoica, “Llms can easily learn to reason from demonstrations structure, not content, is what matters!” arXiv.org, 2025.

[73] G. S. Suvra, S. Chakraborty, A. Reddy, Y. Lu, M. Wang, D. Manocha, F. Huang, M. Ghavamzadeh, and A. S. Bedi, “Does thinking more always help? understanding test-time scaling in reasoning models,” arXiv.org, 2025.

[74] A. P. Gema, A. Hagele, R. Chen, A. Arditi, J. Goldman-Wetzler, K. Fraser-Taliente, H. Sleight, L. Petrini, J. Michael, B. Alex *et al.*, “Inverse scaling in test-time compute,” arXiv.org, 2025.

[75] P. Aggarwal, A. Madaan, Y. Yang *et al.*, “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*, 2023, pp. 12 375–12 396.

[76] Y. Li, P. Yuan, S. Feng, B. Pan, X. Wang, B. Sun, H. Wang, and K. Li, “Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning,” in *The Twelfth International Conference on Learning Representations*, 2024.

[77] C. Huang, L. Huang, J. Leng, J. Liu, and J. Huang, “Efficient test-time scaling via self-calibration,” arXiv.org, 2025.

[78] Y. Wu, Z. Sun, S. Li, S. Welleck, and Y. Yang, “Inference scaling laws: An empirical analysis of compute-optimal inference for llm problem-solving,” in *The Thirteenth International Conference on Learning Representations*, 2025.

[79] X. Wang, Y. Li, S. Feng, P. Yuan, Y. Zhang, J. Shi, C. Tan, B. Pan, Y. Hu, and K. Li, “Every rollout counts: Optimal resource allocation for efficient test-time scaling,” arXiv.org, 2025.

[80] Y. Wang, P. Zhang, S. Huang, B. Yang, Z. Zhang, F. Huang, and R. Wang, “Sampling-efficient test-time scaling: Self-estimating the best-of-n sampling in early decoding,” arXiv.org, 2025.

[81] Y. Wang, J. Wang, R. Chen, X.-Y. Wei, and Q. Li, “Probabilistic optimality for inference-time scaling,” arXiv.org, 2025.

[82] K. Zhang, S. Zhou, D. Wang, W. Y. Wang, and L. Li, “Scaling llm inference efficiently with optimized sample compute allocation,” in *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. Albuquerque, New Mexico: Association for Computational Linguistics, Apr. 2025, pp. 7959–7973.

[83] P. B. Chen, Y. Zhang, D. Roth, S. Madden, J. Andreas, and M. Cafarella, “Log-augmented generation: Scaling test-time reasoning with reusable computation,” arXiv.org, 2025.

[84] J. Song, D. Jo, Y. Kim, and J.-J. Kim, “Reasoning path compression: Compressing generation trajectories for efficient llm reasoning,” arXiv.org, 2025.

[85] Y. Ding, W. Jiang, S. Liu, Y. Jing, J. Guo, Y. Wang, J. Zhang, Z. Wang, Z. Liu, B. Du, X. Liu, and D. Tao, “Dynamic parallel tree search for efficient llm reasoning,” arXiv.org, 2025.

[86] Z. Wang, J. Wu, Y. Lai, C. Zhang, and D. Zhou, “Seed: Accelerating reasoning tree construction via scheduled speculative decoding,” in *Proceedings of the 31st International Conference on Computational Linguistics*, 2025, pp. 4920–4937.

[87] R. Pan, Y. Dai, Z. Zhang, G. Oliaro, Z. Jia, and R. Netravali, “Specreason: Fast and accurate inference-time compute via speculative reasoning,” arXiv.org, 2025.

[88] Z. Ye, L. Chen, R. Lai, W. Lin, Y. Zhang, S. Wang, T. Chen, B. Kasikci, V. Grover, A. Krishnamurthy *et al.*, “Flashinfer: Efficient and customizable attention engine for llm inference serving,” in *Eighth Conference on Machine Learning and Systems*, 2025.

[89] T. Zadouri, H. Strauss, and T. Dao, “Hardware-efficient attention for fast decoding,” arXiv.org, 2025.

[90] X. Miao, G. Oliaro, Z. Zhang, X. Cheng, Z. Wang, Z. Zhang, R. Y. Y. Wong, A. Zhu, L. Yang, X. Shi, C. Shi, Z. Chen, D. Arfeen, R. Abhyankar, and Z. Jia, “Specinfer: Accelerating large language model serving with tree-based speculative inference and verification,” in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, 2024, p. 932–949.

[91] A. Lancucki, K. Staniszewski, P. Nawrot, and E. M. Ponti, “Inference-time hyper-scaling with kv cache compression,” arXiv.org, 2025.

[92] G.-I. Yu, J. S. Jeong, G.-W. Kim, S. Kim, and B.-G. Chun, “Orca: A distributed serving system for Transformer-Based generative models,” in *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, Jul. 2022, pp. 521–538.

[93] R. Qin, Z. Li, W. He, M. Zhang, Y. Wu, W. Zheng, and X. Xu, “Mooncake: A kvcache-centric disaggregated architecture for llm serving,” arXiv.org, 2025.

[94] I. Gim, G. Chen, S.-s. Lee, N. Sarda, A. Khandelwal, and L. Zhong, “Prompt cache: Modular attention reuse for low-latency inference,” *Proceedings of Machine Learning and Systems*, vol. 6, pp. 325–338, 2024.

[95] R. Pan, Z. Wang, Z. Jia, C. Karakus, L. Zancato, T. Dao, Y. Wang, and R. Netravali, “Marconi: Prefix caching for the era of hybrid llms,” in *Eighth Conference on Machine Learning and Systems*, 2025.

[96] Y. Zhu, A. Falahati, D. H. Yang, and M. M. Amiri, “Sentencekv: Efficient llm inference via sentence-level semantic kv caching,” arXiv.org, 2025.

[97] R. Sadhukhan, Z. Chen, H. Zheng, Y. Zhou, E. Strubell, and B. Chen, “Kinetics: Rethinking test-time scaling laws,” arXiv.org, 2025.

[98] J. Juravsky, B. Brown, R. Ehrlich, D. Y. Fu, C. Ré, and A. Mirhoseini, “Hydragen: High-throughput llm inference with shared prefixes,” arXiv.org, 2024.

[99] L. Ye, Z. Tao, Y. Huang, and Y. Li, “Chunkattention: Efficient self-attention with prefix-aware kv cache and two-phase partition,” in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 11 608–11 620.

[100] L. Zhu, X. Wang, W. Zhang, and R. Lau, “Relayattention for efficient large language model serving with long system prompts,” in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 4945–4957.

[101] Z. Pan, Y. Ding, Y. Guan, Z. Wang, Z. Yu, X. Tang, Y. Wang, and Y. Ding, “Fasttree: Optimizing attention kernel and runtime for tree-structured llm inference,” in *Eighth Conference on Machine Learning and Systems*, 2025.

[102] Z. Wang, R. Ning, C. Fang, Z. Zhang, X. Lin, S. Ma, M. Zhou, X. Li, Z. Wang, C. Huan, R. Gu, K. Yang, G. Chen, S. Zhong, and C. Tian, “Flashforge: Ultra-efficient prefix-aware attention for llm decoding,” arXiv.org, 2025.

Appendix

A Datasets, Evaluation, and Verification

A.1 Datasets and Evaluation Methods

Mathematical reasoning. We use AIME 2024, AIME 2025, and MATH500 [14] to evaluate mathematical reasoning. AIME is a high school mathematics competition that features 30 challenging problems each year, and we include all 30 problems from both the 2024 and 2025 editions. MATH500 is a subset of the MATH benchmark comprising challenging problems across multiple topics; we select all problems of difficulty level 5, which is the highest level in MATH500, yielding 134 questions. For evaluation, we prompt the model to place its final answer inside a `\boxed{}` expression. For AIME, where answers are integers, the boxed content can be directly parsed and compared to the reference. For MATH500, where boxed expressions can be more complex, we use `sympy` to check mathematical equivalence between the predicted and reference answers.

Code generation. We use LiveCodeBench v6 [15], which consists of recently collected programming problems from AtCoder and LeetCode. Version 6 covers 175 problems released between January 2025 and May 2025. To evaluate model outputs, we run each generated program against a large set of private test cases, and a solution is considered correct only if it passes all test cases.

Social and logical reasoning. Theory-of-mind (ToM) refers to the ability to infer others' mental states such as beliefs [66, 67]. A common ToM evaluation format is the unexpected transfer task, which can be viewed as a form of commonsense dynamic logical reasoning [68]. For example, in a multi-step story: Alice and Bob are in a room with a chocolate in a box; after Alice leaves, Bob moves the chocolate to the table. A correct solver should infer that Alice still believes the chocolate is in the box. Hi-ToM [16] contains 200 problems that evaluate multi-agent ToM reasoning over long, temporally structured scenarios. To evaluate model outputs, we generate process-level labels, and a prediction is considered correct only if all beliefs are correctly inferred at every step.

A.2 Evaluation Prompts

The prompts for AIME, MATH500, and LiveCodeBench are shown in Figure 7, Figure 8, and Figure 9, respectively. For Hi-ToM, we use the same one-shot prompt format as in **(author?)** [68].

Prompt for AIME

```
Please reason step by step, and put your final answer within
\boxed{}.

{Question}
```

Figure 7: Prompt used for AIME.

Prompt for MATH500

```
Answer the following math question step by step, given in LaTeX
    format, clearly and concisely, and present the final answer as
    \boxed{x}, where X is the fully simplified solution.

Example:
Question: \int_0^1 (3x^2 + 2x) \,dx
Solution: \int (3x^2 + 2x) \,dx = x^3 + x^2 + C
Evaluating from 0 to 1: (1^3 + 1^2) - (0^3 + 0^2) = 1 + 1 - 0 = 2

\boxed{2}

Now, solve the following question step by step.

{Question}
```

Figure 8: Prompt used for MATH500.

Prompt for LiveCodeBench

~~~
You are an expert Python programmer.
You will be given a programming problem and must generate a correct
    Python solution that matches the specification and passes all
    tests.

{Question}

Format:
You will use the following starter code to write the solution
and enclose your code within backticks.

```python
class Solution:
    def solve(self, ...):
        pass

Answer:
~~~

Figure 9: Prompt used for LiveCodeBench.

A.3 Verification Pipeline

For AIME, we apply an additional reasoning-trace verification step to reduce the risk of spurious correctness. Specifically, for every model and every AIME problem, if among 1,024 sampled traces the number of correct answers is ≤ 32 (i.e., Avg@1024 $\leq 3\%$), we perform gpt-5-assisted validation. Gpt-5 is provided with multiple human-written reference solutions and asked to judge whether each model-generated reasoning trace is logically valid and leads to the correct answer. Only traces judged as correct are retained; all others are filtered out. The validation prompt is shown in Figure 10.

B LLM Usage

We used GPT-5 as a general-purpose assist tool for language refinement and proofreading. In addition, GPT-5 was employed to assist in the verification of AIME reasoning traces, as described in Appendix A.

C Algorithm

The algorithm for the efficient temperature scaling can be found in Algorithm 1.

D Extended Related Work

Single-trace TTS. In single-trace TTS, the goal is to encourage deeper and more deliberate reasoning within a single inference path [1, 69]. This can be achieved by RL [2, 3, 70, 71, 18] or by distilling reasoning traces from a stronger teacher model [72]. Compared to single-trace approaches, multi-trace TTS has been shown to produce more stable results [73, 74]. This work focuses on the multi-trace setting.

Efficient algorithms for TTS. One line of work focuses on resource allocation, aiming to allocate more computation to difficult examples and less to easier ones [75–82, 24]. Another line of work incorporates system-level techniques into TTS, such as compressing or reusing the KV cache to avoid redundant computation [83–85, 9], or integrating with speculative decoding to reduce latency [86, 87]. These methods improve test-time efficiency by combining better search with system-level optimizations.

Validation Prompt for AIME

```
You are a rigorous grading instructor for math contest solutions.
You are given the original problem text, the student's full written
    solution (whose final numeric answer is known to be correct),
    and one or more reference solutions (provided only as guidance;
    exact verbatim matching is NOT required).

Your task is to judge whether the student's *reasoning process* is
    logically valid and self-contained, instead of merely checking
    the final answer.

[Original problem]
{ProblemText}

[Student's full solution]
{ModelTrace}

[Reference solution(s) - for guidance only]
Solution 1: ...
Solution 2: ...
Solution 3: ...

Judging criteria:
1) Are key derivations justified with sufficient intermediate steps,
     without circular reasoning or illicit assumptions?
2) Minor arithmetic/notation slips that are explicitly corrected
    later and do not affect the logic may still be acceptable.
3) If there is a fundamental flaw (misused theorem, false equality,
    missing essential conditions) such that the correct final answer
     could be a coincidence, the solution should be judged incorrect
    .
4) Different approaches from the reference are permitted as long as
    the argument forms a logically sound and complete proof.
Please reason step by step before you decide.

Output requirement:
After your analysis, the VERY LAST LINE must be exactly one of the
    following:
[CORRECT]
or
[INCORRECT]
Do not output anything after that final line.
```

Figure 10: Validation prompt used for gpt-5-assisted verification on AIME problems.

Efficient serving system for TTS. Recent work has proposed various serving systems to reduce decoding latency. These include optimized attention kernels [88, 89], speculative decoding methods [90], and KV cache compression techniques [91]. Other efforts improve system throughput via techniques like continuous batching [92] and separating the prefilling and decoding stages [93].

Parallel decoding-based TTS can be viewed as a tree-structured decoding problem, where a shared prefix is expanded into multiple completions. In this setting, many queries access the same KV cache from the shared prefix, making efficient KV cache management a key challenge [94–97]. vLLM addresses this by introducing PagedAttention [10], which reduces memory usage and improves throughput. Building on this, SGLang proposes RadixAttention [11], allowing programmatic control over KV cache reuse. In addition, to address the I/O overhead of reading and writing the KV cache, recent works [98–102] develop new methods to more efficiently compute both the shared prefix and its multiple completions.

Algorithm 1 Efficient Temperature Scaling with Two-Stage Voting

```
Require: Questions $\mathcal{Q}$; temperatures $\mathcal{T} = \{T_1, \ldots, T_M\}$; max rounds $R$; intra-temp threshold
    $\tau_{\text{intra}}$; cross-temp threshold $\tau_{\text{cross}}$; sampler $\textsc{Sample}(q, T)$; extractor $\textsc{Ans}(y)$
Ensure: For each $q \in \mathcal{Q}$, per-temperature pools $\{\mathcal{S}_{q,T}\}$ and their answers
for all $q \in \mathcal{Q}$ do
    for all $T \in \mathcal{T}$ do
        $\mathcal{S}_{q,T} \leftarrow \emptyset$                          ▷ trace pool at temperature $T$
        $\mathcal{A}_{q,T} \leftarrow \emptyset$                          ▷ multiset of answers for $T$
    end for
    $\textsc{active}(q) \leftarrow$ true
end for
for $r \leftarrow 1$ to $R$ do
    for all $q \in \mathcal{Q}$ with $\textsc{active}(q) =$ true do               ▷ Round-$r$ sampling
        for all $T \in \mathcal{T}$ do
            $y \leftarrow \textsc{Sample}(q, T)$
            $a \leftarrow \textsc{Ans}(y)$
            Append $(T, y, a)$ to $\mathcal{S}_{q,T}$; insert $a$ into $\mathcal{A}_{q,T}$
        end for
                                                  ▷ Stage 1: intra-temperature voting
        *all_passed* $\leftarrow$ true;     $\mathcal{V}_q \leftarrow \emptyset$                ▷ $\mathcal{V}_q$: one vote per $T$
        for all $T \in \mathcal{T}$ do
            Build $h_{q,T}(a) \leftarrow \#\{a' \in \mathcal{A}_{q,T} : a' = a\}$
            $v_{\max}^{(T)}(q) \leftarrow \max_a h_{q,T}(a)$
            if $v_{\max}^{(T)}(q) < \tau_{\text{intra}}$ then
                *all_passed* $\leftarrow$ false                  ▷ this $T$ not confident yet
            else
                $a_T \leftarrow \arg\max_a h_{q,T}(a)$                ▷ temperature-$T$ majority
                Append $a_T$ to $\mathcal{V}_q$                     ▷ one vote from this $T$
            end if
        end for
        if *all_passed* = false then
            continue                      ▷ skip cross-temp vote; keep sampling next round
        end if
                                                  ▷ Stage 2: cross-temperature voting
        Build cross-temp tally $H_q(b) \leftarrow \#\{a_T \in \mathcal{V}_q : a_T = b\}$
        $V_{\max}(q) \leftarrow \max_b H_q(b)$
        if $V_{\max}(q) \geq \tau_{\text{cross}}$ then
            $\textsc{active}(q) \leftarrow$ false                        ▷ early exit for $q$
        end if
    end for
    if $\textsc{AllInactive}(\mathcal{Q})$ then
        break
    end if
end for
return $\{\mathcal{S}_{q,T} \mid q \in \mathcal{Q},\ T \in \mathcal{T}\}$          ▷ downstream verifier/BoN picks final answers
```