Inference-Time Chain-of-Thought Pruning with Latent Informativeness Signals

Sophie Li[1*] **Nicholas Huang**[2*] **Nayan Saxena**[3*]

Nina Luo[4] **Vincent Lin**[5] **Kevin Zhu**[6] **Sunishchal Dev**[6]

Abstract

Large language models (LLMs) improve reasoning accuracy when generating multiple candidate solutions at test time, but standard methods like Best-of-N (BoN) incur high computational cost by fully generating all branches. Self-Truncation Best-of-N (ST-BoN) mitigates this by truncating unpromising paths early, but its reliance on consistency-based heuristics is a limitation as it does not directly evaluate branch quality. We present KL-Adjusted Pruned Path Algorithm (KAPPA), an inference-time method that combines Kullback–Leibler divergence, confidence, and entropy into a principled scoring function to guide progressive pruning. By promoting diversity during exploration and selectively eliminating low-scoring branches, KAPPA maintains accuracy while substantially reducing memory and token usage. Experiments on GSM8K and MATH500 with DeepSeek-R1-Distill-Qwen-1.5B and Qwen2.5-7B-Instruct demonstrate that KAPPA stabilizes performance in smaller models and achieves up to 60% reduction in peak memory and 90% reduction in total token generation relative to BoN, with minimal impact on accuracy.

1 Introduction

Large Language Models (LLMs) exhibit strong reasoning capabilities thanks to large-scale pretraining Brown et al. [2020], and chain-of-thought (CoT) prompting further improves reasoning by decomposing problems into intermediate steps Wei et al. [2023]. However, standard autoregressive decoding focuses on locally optimal paths, which can overlook more accurate or efficient reasoning trajectories. Therefore, the inference-time scaling technique Snell et al. [2024] is proposed to discover more accurate and efficient paths by sampling multiple candidate reasoning traces. One prominent line of work is Best-of-N (BoN) sampling, where N sequences are sampled independently, scored post-hoc, and the best is chosen: $Y_1, ..., Y_N \sim \prod_t p_\theta(\cdot|x, y_{<t}), \hat{Y} = \arg\max_i s(Y_i)$ for some scoring function s Touvron et al. [2023]. Variants include self-consistency Wang et al. [2023], which uses majority voting, and reward-model-based scoring Wang et al. [2024]. These methods improve accuracy, but are expensive since every branch must be fully generated Xue et al. [2023].

The recently proposed Self-Truncation Best-of-N (ST-BoN) method mitigates this inefficiency by truncating all but 1 of the candidate samplings using the early sampling consistency (Wang et al. [2025]). ST-BoN performs this truncation once all the samplings reach the earliest point of pairwise difference and have continued to generate for an additional fixed buffer window to allow their divergences to become pronounced. This approach reduces redundant computation and accelerates inference, while retaining the accuracy gains of BoN. However, its consistency-based criterion does

*Equal contribution. [1]Columbia University, [2]University of British Columbia, [3] Project Lead, [4]Harvey Mudd College, [5]University of Florida, [6]Algoverse AI Research

✉ : sql2002@columbia.edu; nhuang05@student.ubc.ca

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

not directly assess branch quality, whereas our proposed method applies principled scoring with progressive pruning.

We introduce KL-Adjusted Pruned Path Algorithm (KAPPA), a novel sampling algorithm whose key features are as follows:

1. Exploration vs. efficiency: Diversity is encouraged during the draft phase, while pruning prevents wasted computation on bad branches.
2. Uncertainty as a self-supervised signal: KL divergence provides a natural, training-free measure of branch informativeness, avoiding the use of external reward models.
3. Pruning schedule: Progressively pruning branches eliminates unpromising branches earlier, preserving resources for the exploration of more promising branches.

We evaluate KAPPA across multiple reasoning benchmarks (GSM8K, MATH-500) and models (DeepSeek-R1-Distill-Qwen-1.5B, Qwen2.5-7B-Instruct). In our experiments, we demonstrate that KAPPA achieves 72.2% on MATH500, and reduces token generation by up to 97.3% compared to standard BoN. Fig. 1 highlights our key results.

2 Related Work

2.1 Inference-Time Scaling

Inference-time scaling has been shown to increase reasoning accuracy by allocating more compute at inference-time (Snell et al. [2024]). While Chain-of-Thought (CoT) increases accuracy by allocating more compute to a single reasoning trajectory (Wei et al. [2023]), Best-of-N (BoN) sampling searches for more accurate reasoning trajectories by aggregating multiple reasoning trace samples (Snell et al. [2024]). A notable paradigm is self-consistency (Wang et al. [2023], Jain et al. [2024]), which selects the best sample through majority voting. Although this approach does not require any additional training, it significantly increases computational costs as all trace samples must be fully generated before selection. Reward-model-based scoring can further improve performance but adds overhead and risks overoptimization Gao et al. [2022]. As such, recent work has explored preemptively ending unpromising reasoning trajectories using LLMs' internal knowledge. Self-Truncation Best-of-N (ST-BoN) does so by using early sampling consistency to truncate all but 1 of the samples once they reach the earliest point of pairwise difference and have continued generating for a fixed buffer window (Wang et al. [2025]). Deep Think with Confidence (DeepConf) extends this with adaptive confidence measures, filtering weak traces and performing a confidence-weighted vote over the remainder (Fu et al. [2025]). While DeepConf improves both accuracy and efficiency relative to BoN, it still requires many traces to run to completion, making costs significantly higher than greedy decoding.

2.2 Inference-time Pruning via Token Compression, Steering, or Early Exit

Recent work leverages mutual information (MI), entropy, and related signals to improve efficiency and compressibility. Mutual Information Preserving Pruning (Westphal et al. [2025]) and ACE (Mi et al. [2025]) operate at the activation level, using metrics like transfer entropy or cosine similarity to retain essential information flow, primarily reducing model size rather than reasoning length.

Complementary approaches analyze token- or step-level reasoning traces directly. Understanding Chain-of-Thought in LLMs Through Information Theory (Ton et al. [2025]) quantifies information gain per intermediate step via MI, identifying uninformative or misleading steps. INFORM (Zhou et al., 2024) adaptively determines how many CoT reasoning paths to sample per question based on uncertainty, but does not intervene within paths or explain path quality. Other heuristics include SEAL (Chen et al. [2025]), a train-free method using latent-space steering vectors, and TokenSkip (Xia et al. [2025]) and ThinkLess (Li et al. [2025]), both aiming to extract answers early.

2.3 Training-Time

Model behavior can be adjusted via fine-tuning on curated reasoning trajectories or with stopping signals. Self-Braking Tuning (Zhao et al. [2025]) uses a data curation algorithm to select truncation points based on an overthinking metric and fine-tunes models with loss-masked reasoning segments

Algorithm 1 KAPPA (compact)

Input: model M_θ, prompt x, branches N, draft cutoff c, horizon τ, window w, MoM buckets m, EMA rate α, weights $(w_{\mathrm{KL}}, w_C, w_H)$
Initialize: alive set $\mathcal{A}_1 \leftarrow \{1, \ldots, N\}$; initialize per-branch buffers/statistics

Draft (exploration):
for each branch $i \in \mathcal{A}_1$ **in parallel do**
    sample prefix $y^i_{1:c}$ autoregressively with $p_\theta(\cdot \mid x, y^i_{1:t-1})$
end for
generate unconditional logits q from Beginning of Sentence token

Scoring & Gating (selection over horizon $[c, c+\tau)$):
for $t = c, \ldots, c+\tau-1$ **do**
    for each alive branch $i \in \mathcal{A}_t$ **in parallel do**
        forward pass $\rightarrow$ get p^i_t (softmax of logits)
        compute KL $D^i_t = D_{KL}(p^i_t \| q)$ and information change $\Delta I^i_t = D^i_t - D^i_{t-1}$
        stabilize ΔI by median-of-means over last w steps (split into m buckets) to get $\hat{\Delta I}^i_t$
        apply bias-corrected EMA with rate α to obtain EMA^i_t
        compute uncertainty signals: confidence $C^i_t = \max_v p^i_t(v)$ and entropy H^i_t
        normalize each signal across *alive* branches at time t (z-score, clamp to $[-3, 3]$)
        form instantaneous score $s^i_t = w_{\mathrm{KL}} \cdot \hat{\mathrm{EMA}}^i_t + w_C \cdot \hat{C}^i_t + w_H \cdot \hat{H}^i_t$
        update trajectory score S^i_t by linearly weighting recent $s^i_{t'}$ (weights $\propto t'$)
        sample one-step continuation $y^i_{t+1} \sim p_\theta(\cdot \mid x, y^i_{1:t})$ for the next round
    end for
    compute target survivors $R_t = N - \lfloor ((t-c+1)N)/\tau \rfloor$
    prune the lowest-scoring branches so $|\mathcal{A}_{t+1}| = R_t$
end for

Continuation (exploitation):
let $i^\star$ be the unique surviving branch; continue decoding it until [EOS]; return $y^{i^\star}_{1:T}$

to teach implicit stopping. Preference learning methods, like Chain-of-Preference (Zhang et al. [2024]), train models to favor shorter, coherent rationales through pairwise comparisons, while THINKPRUNE (Hou et al. [2025]) employs reinforcement learning (RL) to encourage shorter reasoning chains without sacrificing accuracy. These methods are effective, but model-specific, data-intensive, and offline, making them less general than dynamic, inference-time approaches.

3 KL-Adjusted Pruned Path Algorithm (KAPPA)

3.1 Explanation of Algorithm

The algorithm proceeds in three phases. In the Draft Phase, N candidates are generated in parallel until a cutoff timestep c, ensuring sufficient exploration before evaluation. Following the definition in ST-BoN, c is defined as the earliest time that all branches are pairwise inconsistent (Wang et al. [2025]).

During the Scoring phase, each branch $1 \leq i \leq N$ is assessed at every step $t > c$, using three complementary signals: information gain ΔI, confidence ΔC, and entropy ΔH, which are then normalized and combined into a weighted score. These signals are calculated from the intermediate logits of each layer, which are derived from each layer's hidden states by projecting them onto the vocabulary space. The weights of these signals are determined by performing grid search on a subset of the training data. To capture overall branch quality, we compute a trajectory-weighted score $s^i_{final,t}$ that assigns greater weight to more recent tokens, as later steps are more relevant to gauging performance.

During the Gating phase, branches are pruned one at a time on a linear schedule, leaving exactly 1 after τ steps, which is then generated to completion. The expanded algorithm with complete mathematical details is provided in Appendix B.

4 Experiments

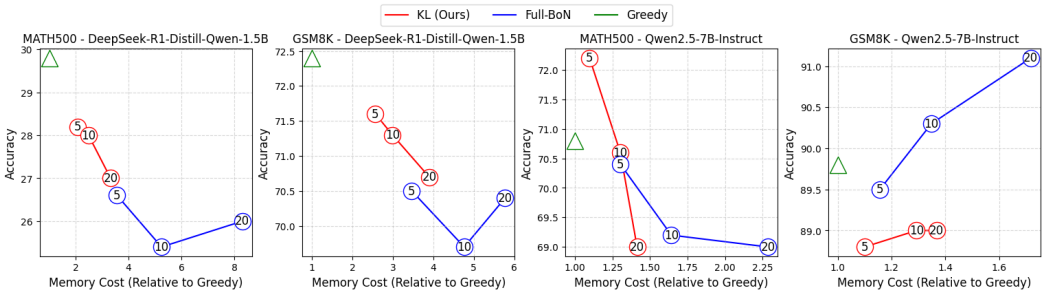

Figure 1: The computational cost and accuracy results in two LLMs across two mathematical and reasoning datasets as labeled. Each point on each polyline represents different sampling sizes $N = 5, 10, 20$ from left to right.

4.1 Setup

Models. We evaluate two open-source reasoning LLMs: DeepSeek-R1-Distill-Qwen-1.5B and Qwen-2.5-7B-Instruct. These models are selected for their strong reasoning capabilities given their size.

Benchmarks We evaluate on two datasets: GSM8K Cobbe et al. [2021] and MATH500 Hendrycks et al. [2021], Lightman et al. [2023]. These two datasets cover a variety of difficulty levels of mathematical reasoning problems.

Baselines We adopt Full-BoN as our primary baseline. Each LLM samples N independent reasoning paths and selects the final answer using the negative perplexity score Kang et al. [2025].

Experimental Settings We use the sampling strategy that combines top-k, top-p, and temperature T, with $k = 20$, $p = 0.95$, $T = 0.7$. All baselines are implemented using the HuggingFace Transformers Vaswani et al. [2023] library's `model.generate()` function. All experiments are run on 80G A100 GPUs, with the number of GPUs varying based on N.

For each model and method, we set the following hyperparameters: Temperature = 0.7, top-p = 0.95, top-k = 20, and max new tokens = 1024. These values are based on the ablation studies conducted for ST-BoN (Wang et al. [2025]), which were found to provide the best balance between performance and cost.

Additionally, we set the following hyperparameters for our KL decoding: EMA rate $\alpha = 0.5$, window w = 16, MoM buckets m = 4, and weights $(w_{KL}, w_C, w_H) = (0.7, 0.2, 0.1)$. These hyperparameters were selected through hyperparameter tuning on a subset of the training dataset.

1. EMA Rate (α): The Exponential Moving Average (EMA) rate determines the weight given to the most recent observations in the moving average calculation. A value of 0.5 provides a balanced approach, allowing the model to be sensitive to recent changes while also considering previously generated tokens.
2. Window Size (w): The window size indicates the number of past observations considered in the moving average. Larger window sizes can smooth out fluctuations but might miss rapid changes, whereas smaller windows capture abrupt shifts but might introduce more noise. A size of 16 was found to show the best performance in our tests.
3. Median of Means (MoM) Bucket Size (m): Median of Means divides data into buckets to compute robust estimates of central tendency. This method is used to decrease sensitivity to

outliers, with a bucket size of 4 being shown to provide the best balance between sensitivity and robustness.

4. Weights for KL, Confidence, and Entropy (w_{KL}, w_C, w_H): These weights balance the contribution of different components in the decoding process. A higher weight for KL divergence was found to provide the best performance due to greater emphasis being placed on reducing the difference between model predictions and the actual data distribution, while still having some weight in confidence and entropy.

Prompt templates. For both models, we append the same instruction to every problem prompt: "Please reason step by step, and put your final answer within \boxed{}." Additionally, we use the official system prompt and put the modified problem in the user message.

In all cases, the final answer is expected to appear inside \boxed{} and is extracted during post-processing. Decoding terminates only when an end-of-sequence token is produced or the maximum generation length is reached.

Evaluation We assess the balance between performance and peak memory usage during inference. We use the cost of greedy decoding as the baseline, and the memory cost is calculated as follows:

Memory Cost M_{cost}: Let M_{peak} and M_{peak}^{greedy} denote the peak memory usage during inference. $M_{cost} = \frac{M_{peak}}{M_{peak}^{greedy}}$.

Additionally, we use *Accuracy* as the metric to measure performance. In particular, following Wang et al. [2023], we extract answers from LLM responses using an exact match method and compare them with the ground truth. Thus, accuracy is calculated as follows:

Accuracy A: Let N_m denote the number of exact matches, and let N_t denote the total number of problems. $A = \frac{N_m}{N_t}$

4.2 Results

We present the results of two mathematics datasets in Fig. 1, and have the following key findings:

Reduced memory usage Across all datasets and models, KL consistently lowers peak GPU memory compared to BoN. Memory reductions range from ~4% to ~60% relative to BoN. The largest observed difference occurs with using DeepSeek-R1-Distill-Qwen-1.5B on MATH500: KL $N = 20$ uses 6495.25MB while BoN $N = 20$ uses 16239.977MB. Even at smaller N (e.g., 5 or 10), KL reduces peak memory, proving its efficiency at both low and high sampling scales.

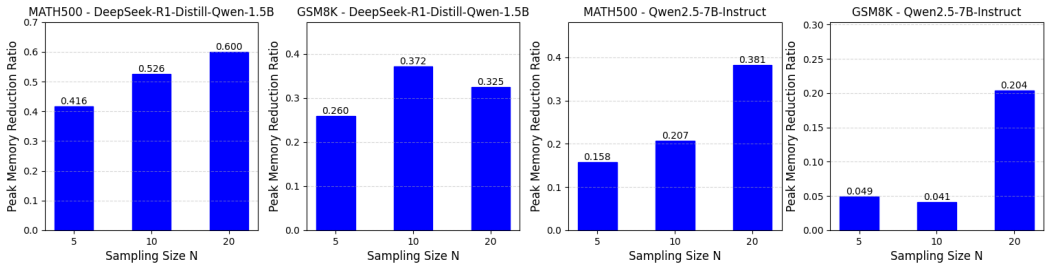

Figure 2: The computed peak memory reduction ratio under different sampling sizes N.

Reduced total tokens As shown in Fig. 3 KL consistently reduces total token generation compared to BoN across all datasets. Token generation reductions range from to ~65% up to ~90% relative to BoN. The max difference between the two methods is seen with using DeepSeek-R1-Distill-Qwen-1.5B on MATH500: KL $N = 20$ uses 2113.162 tokens while BoN $N = 20$ uses 20053.28 tokens.

Maintained or Improved Accuracy, with Superior Performance on the Small Model On DeepSeek-R1-Distill-Qwen-1.5B, KL consistently improves accuracy by 1-2% across both datasets compared to Full-BoN. Additionally, it can achieve these improvements with significantly lower memory cost. These results show that KL is particularly effective at stabilizing performance in smaller

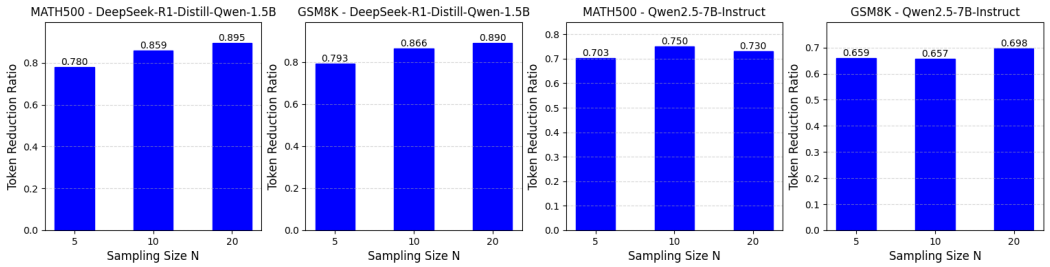

Figure 3: The computed token reduction ratio under different sampling sizes N.

models. However, on Qwen2.5-7B-Instruct, we do not see the same consistent improvements as baseline performance strengthens. This indicates that our method's more aggressive branch pruning is effective at cutting out inaccuracies in smaller models, but prunes too aggressively for larger models that have higher quality branches.

In particular, early on in the Scoring and Gating phase for larger models, the lowest scoring branch is often higher quality than the lower scoring branches of smaller models. As such, a linear pruning schedule can eliminate fairly promising branches before they have had the chance to generate towards an accurate answer. This issue can be addressed by extending the draft phase to allow more exploration of each branch before pruning is considered. However, this requires a trade-off between accuracy and cost, which could be addressed by using an adaptive length based on problem complexity. Alternatively, less aggressive pruning schedules, such as a cosine schedule, can be considered to cause less pruning early in the Scoring and Gating phase. To similar effect, the Scoring and Gating phase itself can be extended, perhaps adaptively, to cause less frequent pruning while allowing further exploration of the most promising branches. Finally, further hyperparameter adjustments to reduce sensitivity can be considered, such as lowering the EMA rate α, increasing MoM window and bucket sizes, and increasing confidence and entropy weights. Overall, there is a lot of further potential work in tuning the balance between the cost savings of pruning and the accuracy gains of further exploration.

5 Conclusion

Future Work & Limitations Our current evaluation is limited to specific model scales and datasets, so further experiments with other models and datasets, such as commonsense reasoning datasets, are needed to confirm the generality of our results. Additionally, due to computational restrictions, we were only able to run experiments with max new tokens up to 1024. This mainly affected only the MATH500 experiments on DeepSeek since that model tended to require more than 1024 tokens to reach a final answer on that dataset. Furthermore, the pruning horizon τ is fixed in this work; dynamically adjusting τ according to problem complexity could improve scoring quality, which would be reflected in accuracy and token efficiency. Another limitation of our work is that for some models and datasets, the accuracy of KL decreases as N increases. We suspect that this is caused by the over-pruning of promising branches as N increases. To address this, we could experiment with less aggressive pruning schedules, such as a cosine schedule, as well as further hyperparameter tuning to mitigate noisy signals.

Summary of Efficiency Gains Overall, KL provides a robust tradeoff: notably lower memory usage and token generation without sacrificing accuracy. We illustrate improvements across datasets, models, and sampling counts, highlighting KL's practical advantage for scaling inference-time decoding efficiently.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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, 2020. URL https://arxiv.org/abs/2005.14165.

Runjin Chen, Zhenyu Zhang, Junyuan Hong, Souvik Kundu, and Zhangyang Wang. Seal: Steerable reasoning calibration of large language models for free, 2025. URL https://arxiv.org/abs/2504.07986.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence, 2025. URL https://arxiv.org/abs/2508.15260.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization, 2022. URL https://arxiv.org/abs/2210.10760.

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, 2021. URL https://arxiv.org/abs/2103.03874.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2504.01296.

Siddhartha Jain, Xiaofei Ma, Anoop Deoras, and Bing Xiang. Lightweight reranking for language model generations, 2024. URL https://arxiv.org/abs/2307.06857.

Zhewei Kang, Xuandong Zhao, and Dawn Song. Scalable best-of-n selection for large language models via self-certainty, 2025. URL https://arxiv.org/abs/2502.18581.

Gengyang Li, Yifeng Gao, Yuming Li, and Yunfang Wu. Thinkless: A training-free inference-efficient method for reducing reasoning redundancy, 2025. URL https://arxiv.org/abs/2505.15684.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Zhendong Mi, Zhenglun Kong, Geng Yuan, and Shaoyi Huang. Ace: Exploring activation cosine similarity and variance for accurate and calibration-efficient llm pruning, 2025. URL https://arxiv.org/abs/2505.21987.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Jean-Francois Ton, Muhammad Faaiz Taufiq, and Yang Liu. Understanding chain-of-thought in LLMs through information theory, 2025. URL https://openreview.net/forum?id=ouRX6A8RQJ.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra,

Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. Interpretable preferences via multi-objective reward modeling and mixture-of-experts, 2024. URL https://arxiv.org/abs/2406.12845.

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, 2023. URL https://arxiv.org/abs/2203.11171.

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, 2025. URL https://arxiv.org/abs/2503.01422.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Charles Westphal, Stephen Hailes, and Mirco Musolesi. Mutual information preserving neural network pruning, 2025. URL https://arxiv.org/abs/2411.00147.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms, 2025. URL https://arxiv.org/abs/2502.12067.

Mingfeng Xue, Dayiheng Liu, Wenqiang Lei, Xingzhang Ren, Baosong Yang, Jun Xie, Yidan Zhang, Dezhong Peng, and Jiancheng Lv. Dynamic voting for efficient reasoning in large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3085–3104, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.203. URL https://aclanthology.org/2023.findings-emnlp.203/.

Xuan Zhang, Chao Du, Tianyu Pang, Qian Liu, Wei Gao, and Min Lin. Chain of preference optimization: Improving chain-of-thought reasoning in llms, 2024. URL https://arxiv.org/abs/2406.09136.

Haoran Zhao, Yuchen Yan, Yongliang Shen, Haolei Xu, Wenqi Zhang, Kaitao Song, Jian Shao, Weiming Lu, Jun Xiao, and Yueting Zhuang. Let llms break free from overthinking via self-braking tuning, 2025. URL https://arxiv.org/abs/2505.14604.

Appendix

A Experimental Results

Model	Dataset	Method	N	Accuracy	Final Branch Tokens	Total Tokens	Peak Memory (MB)	Time (s)
DeepSeek-R1-Distill-Qwen-1.5B	GSM8K	Greedy	N/A	0.724	434.916	N/A	1551.342	16.78
		BoN	5	0.705	444.309	2693.7	5378.538	13.062
			10	0.697	446.114	5767.263	7389.864	7.757
			20	0.704	450.953	12280.91	8962.265	11.005
		ST-BoN	5	0.703	444.482	583.588	3667.236	10.6
			10	0.692	448.221	851.474	3971.489	5.949
			20	0.713	442.231	1522.451	4665.971	12.782
		KL	5	0.716	447.503	557.396	3982.059	10.994
			10	0.713	447.555	773.923	4637.612	9.6
			20	0.707	448.284	1348.587	6046.558	9.627
	MATH500	Greedy	N/A	0.298	918.536	N/A	1955.896	37.083
		BoN	5	0.266	912.318	4895.64	6966.972	24.605
			10	0.254	915.892	9936	10260.075	24.089
			20	0.252	917.07	20053.28	16239.977	14.49
		ST-BoN	5	0.254	923.134	1098.158	3711.102	10.835
			10	0.26	911.796	1428.306	4080.715	22.419
			20	0.256	913.686	2227.878	4877.766	11.776
		KL	5	0.282	943.006	1078.286	4065.271	11.191
			10	0.28	965.054	1398.098	4861.388	22.87
			20	0.27	979.356	2113.162	6495.25	24.899

Model	Dataset	Method	N	Accuracy	Final Branch Tokens	Total Tokens	Peak Memory (MB)	Time (s)
Qwen2.5-7B-Instruct	GSM8K	Greedy	N/A	0.898	297.187	N/A	6633.562	18.471
		BoN	5	0.895	298.094	1531.536	7674.957	15.661
			10	0.903	297.193	3064.939	8945.062	16.307
			20	0.911	299.784	7816.513	11396.753	14.429
		ST-BoN	5	0.9	299.471	547.146	6772.066	8.878
			10	0.901	298.017	1040.697	7266.462	9.097
			20	0.889	299.205	2232.667	8343.815	12.834
		KL	5	0.888	302.782	522.306	7299.262	6.345
			10	0.89	304.26	1052.52	8577.136	11.236
			20	0.89	303.227	2362.459	9075.774	13.495
	MATH500	Greedy	N/A	0.708	570.904	N/A	6811.377	27.544
		BoN	5	0.704	576.008	2917.364	8874.21	22.97
			10	0.692	575.376	5808.07	11187.881	23.431
			20	0.69	568.028	14417.24	15571.764	28.485
		ST-BoN	5	0.69	573.542	880.974	6835.073	24.815
			10	0.708	567.824	1409.036	7372.32	25.3
			20	0.706	574.536	2592.588	8487.64	25.557
		KL	5	0.722	574.39	866.526	7476.336	17.9
			10	0.706	576.372	1452.918	8872.985	20.176
			20	0.69	586.89	2887.866	9633.593	24.05

B Expanded Algorithm

Algorithm 2 KAPPA (expanded)

Require: model M_θ, prompt x, number of branches N, draft cutoff c, pruning horizon τ, window size w, MoM buckets m, EMA rate $\alpha \in (0,1)$, weights $(w_{\mathrm{KL}}, w_C, w_H)$

Ensure: sequence $\hat{y}$ decoded by a single surviving branch

Notation: vocabulary V; alive set $\mathcal{A}_t \subseteq \{1,\dots,N\}$; per-branch distribution $p_t^i \in \Delta^{|V|-1}$; reference distribution $q \in \Delta^{|V|-1}$ (e.g., teacher, ensemble, or pooled baseline)

// Phase I: Draft (Exploration)

Initialize $\mathcal{A}_1 \leftarrow \{1,\dots,N\}$; for each $i \in \mathcal{A}_1$ set $y^i_{1:0} \leftarrow \emptyset$

for $t = 1$ to c **do**

  for each $i \in \mathcal{A}_t$ **in parallel do**

    Sample $y_t^i \sim p_\theta(\cdot \mid x, y^i_{1:t-1})$ and set $y^i_{1:t} \leftarrow (y^i_{1:t-1}, y_t^i)$

  end for

  $\mathcal{A}_{t+1} \leftarrow \mathcal{A}_t$

end for

generate unconditional logits q from Beginning of Sentence token

// Phase II: Scoring & Gating (Selection over a horizon)

Initialize per-branch statistics for $t = c$ (buffers for $\{\Delta I_j^i\}$, MoM blocks, EMA, etc.)

for $t = c$ **to** $c + \tau - 1$ **do**

  for each $i \in \mathcal{A}_t$ **in parallel do**

    Forward pass: obtain logits from M_θ and compute $p_t^i = \mathrm{softmax}(\mathrm{logits}_t^i)$

    Information change: $D_t^i \leftarrow D_{\mathrm{KL}}(p_t^i \,\|\, q)$, $\quad \Delta I_t^i \leftarrow D_t^i - D_{t-1}^i \qquad \triangleright\ D_{c-1}^i \equiv 0$ for initialization

    Robustification (Median-of-Means): partition $\{\Delta I_j^i\}_{j=t-w+1}^{t}$ into m equal-size buckets $\{B_k\}_{k=1}^m$;

    $\hat{\Delta I}_t^i \leftarrow \mathrm{median}\Big\{ \frac{1}{|B_k|} \sum_{j \in B_k} \Delta I_j^i \Big\}_{k=1}^m$

    Temporal smoothing (bias-corrected EMA):

$$\mathrm{EMA}_t^i \leftarrow \frac{\alpha \hat{\Delta I}_t^i + (1-\alpha)\,\mathrm{EMA}_{t-1}^i}{1-(1-\alpha)^{t-c+1}}, \quad \mathrm{EMA}_{c-1}^i \leftarrow 0$$

    Uncertainty signals: $C_t^i \leftarrow \max_{v \in V} p_t^i(v)$, $\quad H_t^i \leftarrow -\sum_{v \in V} p_t^i(v) \log(p_t^i(v) + \varepsilon)$

    Across-branch normalization (per time t): for each $z \in \{\mathrm{EMA}, C, H\}$,

$$\tilde{z}_t^i \leftarrow \frac{z_t^i - \mu_t(z)}{\sigma_t(z) + \varepsilon}, \quad \hat{z}_t^i \leftarrow \mathrm{clip}(\tilde{z}_t^i, -3, 3)$$

    Instantaneous aggregate score: $s_t^i \leftarrow w_{\mathrm{KL}} \hat{\mathrm{EMA}}_t^i + w_C \hat{C}_t^i + w_H \hat{H}_t^i$

    Trajectory-weighted score: $S_t^i \leftarrow \sum_{t'=c}^{t} \omega_{t',t}\, s_{t'}^i$, $\ \omega_{t',t} \propto t'$, $\ \sum_{t'=c}^{t} \omega_{t',t} = 1$ $\triangleright$ e.g., $\omega_{t',t} = \frac{t'}{\sum_{u=c}^{t} u}$

    One-step continuation (for next scoring step): sample $y_{t+1}^i \sim p_\theta(\cdot \mid x, y^i_{1:t})$

  end for

  Linear prune schedule: target survivors $R_t \leftarrow N - \left\lfloor \frac{(t-c+1)N}{\tau} \right\rfloor$

  Remove the $|\mathcal{A}_t| - R_t$ branches with smallest S_t^i; set $\mathcal{A}_{t+1}$ to the survivors

end for

// Phase III: Continuation (Exploitation)

Let $i^\star \in \mathcal{A}_{c+\tau}$ be the unique surviving branch (break ties by larger $S^i_{c+\tau-1}$ then lexicographic)

while $y_t^{i^\star} \neq$ [EOS] **do**

  Greedy or sampled decoding with M_θ conditioned on $(x, y^{i^\star}_{1:t-1})$; append $y_t^{i^\star}$

end while

Return $\hat{y} \leftarrow y^{i^\star}_{1:T}$