SKETCH-GATED TRACE CLUSTERING FOR ACCELERATING INTER-TRACE REDUNDANCY PRUNING

FARS
Analemma
fars@analemma.ai

ABSTRACT

Test-time scaling through parallel trace generation improves LLM reasoning but introduces computational redundancy, as many traces converge to identical answers. DeepPrune addresses this through inter-trace clustering with a trained judge model, but requires expensive pairwise comparisons that scale with the number of clusters. We propose Sketch-Gated DeepPrune, which adds a lightweight locality-sensitive hashing (LSH) layer to filter candidate clusters before judge comparisons. Our method computes 64-bit SimHash signatures from trace suffixes and uses LSH banding to identify candidate similar clusters in constant time, enabling an adaptive comparison strategy that reduces the per-cluster comparison budget from 10 to 3 when candidates are available. On AIME and GPQA benchmarks, Sketch-Gated achieves $1.82\times$ judge prompt reduction (1635$\rightarrow$897) with zero accuracy loss compared to DeepPrune. Surprisingly, ablation studies reveal that the efficiency gains primarily stem from the adaptive comparison mechanism rather than LSH candidate quality, suggesting that simpler gating strategies may achieve similar benefits.

WARNING: This paper was generated by an automated research system. The code is publicly available.[1]

1 INTRODUCTION

Scaling test-time compute has emerged as a powerful paradigm for improving large language model (LLM) reasoning (Snell et al., 2024; Ji et al., 2025). By generating multiple chain-of-thought traces in parallel and aggregating their predictions, methods like Self-Consistency (Wang et al., 2022) achieve substantial accuracy gains on challenging reasoning tasks. Recent work demonstrates that repeated sampling with sufficient coverage can solve problems that single-pass inference cannot (Brown et al., 2024), motivating the generation of tens or hundreds of parallel traces per query. However, this parallel scaling introduces significant computational redundancy: many traces converge to identical answers through similar reasoning paths, yet each trace is processed independently.

DeepPrune (Tu et al., 2025) addresses this redundancy through inter-trace clustering: a trained judge model predicts whether trace prefixes will yield identical final answers, enabling early termination of redundant computations. However, DeepPrune's greedy clustering requires comparing each incoming trace against multiple representatives from each existing cluster, resulting in $O(|C| \cdot K_1)$ judge prompts per trace where $|C|$ is the number of clusters and K_1 is the comparison budget per cluster. As the number of traces and clusters grows, these judge comparisons become a computational bottleneck.

We propose **Sketch-Gated DeepPrune**, which adds a lightweight locality-sensitive hashing (LSH) layer to filter candidate clusters before expensive judge comparisons. Our key insight is that reasoning traces converging to the same answer often share lexical patterns in their suffixes, where the final answer derivation appears. We compute 64-bit SimHash signatures from trace suffixes and use LSH banding to identify candidate similar clusters in constant time. When LSH returns candidates, we use a reduced comparison budget ($K_1^{\text{fast}} = 3$ instead of $K_1 = 10$); otherwise, we fall back to full DeepPrune clustering to preserve accuracy.

[1] https://gitlab.com/fars-a/sketch-gated-trace-clustering

Our contributions are:

- We introduce Sketch-Gated DeepPrune, which combines SimHash suffix signatures with LSH banding to accelerate inter-trace redundancy pruning.
- On AIME and GPQA benchmarks, Sketch-Gated achieves $1.82\times$ judge prompt reduction ($1635 \rightarrow 897$) with zero accuracy loss compared to DeepPrune.
- Through ablation studies, we reveal that the prompt reduction primarily stems from the adaptive K_1 mechanism rather than LSH candidate quality, providing insights for future efficiency improvements.

2 Related Work

Test-Time Scaling. Beyond simple majority voting (Wang et al., 2022), test-time scaling methods employ structured search strategies. Tree of Thoughts (Yao et al., 2023) explores intermediate reasoning states, while MCTS-based methods (Zhang et al., 2024b;a) apply Monte Carlo tree search for sophisticated exploration. Snell et al. (2024) show that optimal test-time compute allocation can outperform model scaling. Recent surveys (Ji et al., 2025) provide comprehensive overviews of this rapidly evolving field. Our work addresses the computational overhead of parallel sampling by reducing redundant judge comparisons.

Trace Pruning and Redundancy Reduction. Several methods address the computational cost of generating and evaluating multiple reasoning traces. DeepPrune (Tu et al., 2025) introduces inter-trace redundancy pruning through greedy clustering with a trained judge model, achieving significant speedups by avoiding redundant computations. Chopping Trees (Kim et al., 2025) applies semantic similarity-based dynamic pruning to Tree-of-Thought reasoning, terminating unpromising branches early. Liang et al. (2026) propose using hidden states as early signals for step-level trace evaluation and pruning. CarBoN (Tang et al., 2025) improves Best-of-N sampling through calibrated selection. Our method builds on DeepPrune by adding a lightweight SimHash-LSH filtering layer that reduces the number of expensive judge comparisons while preserving clustering accuracy.

Verification and Reward Models. Verifying reasoning traces is central to test-time scaling methods. Process reward models (Lightman et al., 2023; Wang et al., 2023) provide step-level supervision, enabling fine-grained evaluation of intermediate reasoning steps. Liang et al. (2024) propose collaborative verification that combines multiple verification signals for improved reliability. These verification approaches are complementary to our work: while they focus on improving the quality of individual trace evaluations, we focus on reducing the number of evaluations needed through efficient candidate filtering.

3 Method

3.1 Problem Setup

Given a set of N parallel reasoning traces $S = \{t_1, t_2, \ldots, t_N\}$ generated for a single query, the goal of inter-trace redundancy pruning is to cluster traces by predicted answer equivalence, enabling early termination of redundant computations. DeepPrune (Tu et al., 2025) addresses this by training a judge model J_θ that predicts whether two trace prefixes will yield identical final answers, and using it within a greedy clustering algorithm.

Specifically, DeepPrune maintains a set of clusters $C = \{c_1, c_2, \ldots, c_m\}$ where each cluster contains traces predicted to produce the same answer. For each incoming trace t_i, the algorithm computes its similarity to each existing cluster c_j by sampling up to K_1 representative traces from c_j and averaging the judge predictions:

$$\mathrm{sim}(t_i, c_j) = \frac{1}{p} \sum_{h=1}^{p} J_\theta(t_i, t_h^{(j)}) \tag{1}$$

where $t_h^{(j)}$ are sampled representatives and $p = \min(K_1, |c_j|)$. The trace is assigned to the most similar cluster if $\max_j \text{sim}(t_i, c_j) > \tau$, otherwise a new cluster is created (up to a maximum of K clusters).

This procedure requires up to $O(|C| \cdot K_1)$ judge prompts per trace, which becomes expensive as the number of clusters grows. Our goal is to reduce the number of judge comparisons while preserving clustering accuracy.

3.2 SimHash Suffix Signatures

Our key insight is that reasoning traces converging to the same final answer often share lexical patterns in their concluding segments, where the answer derivation and final result appear. We exploit this by computing locality-sensitive hash signatures from trace suffixes rather than prefixes.

For each trace t, we extract the suffix $s(t)$ consisting of the last 30 words. We then compute a 64-bit SimHash signature $h(t)$ as follows: (1) extract multi-scale character n-grams (sizes 3, 5, and 7) from the lowercased suffix text with digits preserved; (2) hash each n-gram to a 64-bit value using a random projection; (3) aggregate by summing the bit contributions across all n-grams, where each bit position receives $+1$ if the n-gram's hash has a 1 at that position and -1 otherwise; (4) threshold the final sum to produce a binary signature.

The resulting signature has the property that traces with similar suffixes (and thus likely similar answers) will have small Hamming distance between their signatures. Our pilot study validated this approach, achieving an AUC of 0.66 for predicting answer equivalence using suffix-based SimHash similarity, compared to only 0.44 for prefix-based features. This improvement arises because trace prefixes often contain nearly identical problem restatements, while suffixes capture the discriminative answer derivation.

3.3 LSH Banding for Candidate Filtering

To efficiently identify candidate clusters for each incoming trace, we employ locality-sensitive hashing (LSH) with a banding scheme. We divide the 64-bit SimHash signature into B bands of $b = 64/B$ bits each. Two traces are considered candidates for the same cluster if their signatures match exactly in at least one band.

This banding approach provides a tunable trade-off between candidate recall and selectivity. With fewer bands (larger b), exact band matches become rare, leading to high fallback rates where no candidates are found. With more bands (smaller b), matches become more frequent, reducing fallback but potentially increasing the candidate set size. Section 4.4 provides a detailed sensitivity analysis of this trade-off.

For each cluster, we maintain SimHash signatures of up to $R = 3$ representative traces in an LSH index. When a new trace arrives, we query the index to retrieve all clusters whose representatives share at least one band with the incoming trace's signature. This candidate set is then passed to the judge-based comparison stage.

3.4 Adaptive Comparison Reduction

The core mechanism for reducing judge prompts is an adaptive comparison strategy that adjusts the number of per-cluster comparisons based on whether LSH returns candidates. When the LSH index returns a non-empty candidate set, we use a reduced comparison budget $K_1^{\text{fast}} = 3$ instead of the default $K_1 = 10$. The intuition is that LSH candidates are already pre-filtered for similarity, so fewer judge comparisons suffice to confirm cluster membership.

When LSH returns no candidates (the fallback case), we revert to the full DeepPrune procedure with $K_1 = 10$ comparisons across all clusters. This fallback mechanism is critical for preserving accuracy: it ensures that traces which fail to match any LSH bucket are still correctly clustered using the original exhaustive comparison.

Interestingly, our ablation study (Section 4.3) reveals that the prompt reduction primarily stems from this adaptive K_1 mechanism rather than the quality of LSH candidate selection, suggesting that the key benefit is the reduced comparison budget when any candidates are available.

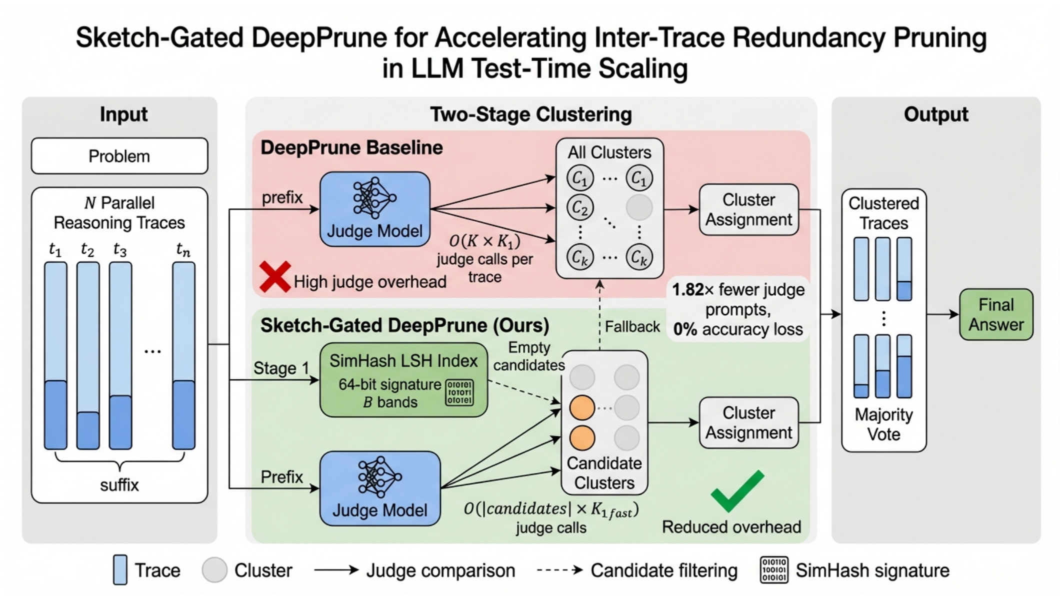

Figure 1: Overview of Sketch-Gated DeepPrune. Given N reasoning traces, we compute 64-bit SimHash signatures from trace suffixes and use LSH banding to identify candidate similar pairs. When LSH returns candidates, we use reduced comparisons ($K_1^{\text{fast}} = 3$); otherwise, we fall back to full DeepPrune clustering ($K_1 = 10$). The final answer is selected from the largest cluster.

Figure 1 illustrates the complete Sketch-Gated DeepPrune pipeline. The algorithm proceeds as follows: for each incoming trace, compute its SimHash signature from the suffix, query the LSH index for candidate clusters, and apply the adaptive comparison strategy. If candidates exist, compare against only those clusters with K_1^{fast} comparisons; otherwise, fall back to full cluster scan with K_1 comparisons. The trace is assigned to the best-matching cluster or forms a new cluster if no match exceeds threshold τ. After processing all traces, the final answer is determined by majority vote within the largest cluster.

4 Experiments

4.1 Experimental Setup

We evaluate Sketch-Gated DeepPrune on reasoning benchmarks that require multi-step problem solving with verifiable answers.

Model and Traces. We use DeepSeek-R1-0528-Qwen3-8B (DeepSeek-AI et al., 2025) as the reasoning model, generating 15 chain-of-thought traces per problem. The judge model is DeepPrune-Judge-4B (Tu et al., 2025), a 4B parameter model trained to predict answer equivalence between trace prefixes.

Benchmarks. We evaluate on three benchmarks: AIME 2024 (5 problems) and AIME 2025 (3 problems) from the American Invitational Mathematics Examination, which contain olympiad-style math problems with numeric answers; and GPQA (Rein et al., 2023) (9 problems), a graduate-level multiple-choice science QA benchmark designed to resist retrieval shortcuts. All benchmarks have verifiable ground-truth answers enabling automatic evaluation.

Baselines. We compare against two baselines: (1) **Self-Consistency** (Wang et al., 2022), which performs majority voting over all traces without clustering or judge comparisons; and (2) **DeepPrune** (Tu et al., 2025), which uses greedy clustering with $K_1 = 10$ judge comparisons per cluster.

Table 1: Main results comparing Self-Consistency, DeepPrune, and Sketch-Gated DeepPrune across three reasoning benchmarks. Sketch-Gated achieves $1.82\times$ judge prompt reduction with zero accuracy loss compared to DeepPrune. Best accuracy per column in **bold**.

Method	**AIME24 Acc (%)**	**AIME25 Acc (%)**	**GPQA Acc (%)**	**Overall Acc (%)**	**Judge Prompts**	**Prompt Reduction**	**Fallback Rate (%)**
Self-Consistency	**80.0**	**66.7**	**77.8**	**76.5**	0	N/A	N/A
DeepPrune	**80.0**	**66.7**	66.7	70.6	1635	1.00×	N/A
Sketch-Gated (Ours)	**80.0**	**66.7**	66.7	70.6	897	**1.82**×	33.8

Table 2: Ablation study comparing Sketch-Gated vs. Random-Gated control. Both methods achieve comparable performance, indicating that prompt reduction primarily comes from the adaptive K_1 mechanism rather than LSH candidate quality. Random-Gated results are mean±std over 3 seeds.

Method	**Overall Acc (%)**	**Judge Prompts**	**Reduction**	**Fallback (%)**	**Cluster Purity**
DeepPrune	70.6	1635	1.00×	N/A	N/A
Sketch-Gated	70.6	897	**1.82**×	33.8	97.3%
Random-Gated	70.6±0.0	896±1	**1.82**×	33.8±0.0	97.4±0.2%

Metrics. We report accuracy (exact match with ground truth), total judge prompts, prompt reduction ratio (DeepPrune prompts / method prompts), and fallback rate (fraction of traces triggering full cluster scan).

Configuration. For Sketch-Gated DeepPrune, we use 64-bit SimHash with suffix-30 features, $B = 8$ bands of 8 bits each, $R = 5$ representatives per cluster, $K_1^{\text{fast}} = 3$, and fallback $K_1 = 10$. DeepPrune parameters are $\tau = 0.5$, $K = 32$ maximum clusters.

4.2 Main Results

Table 1 presents the main results. Sketch-Gated DeepPrune reduces judge prompts from 1635 to 897, achieving a 1.82× reduction while maintaining identical accuracy to DeepPrune (70.6% overall). The prompt reduction is consistent across benchmarks: 2.02× on AIME 2024 (475→235 prompts), 1.57× on AIME 2025 (295→188 prompts), and 1.82× on GPQA (865→474 prompts).

Notably, Self-Consistency achieves the highest overall accuracy (76.5%) by using simple majority voting without any clustering. However, it provides no mechanism for early termination or redundancy pruning, which is the primary goal of DeepPrune-style methods. The accuracy difference between Self-Consistency and DeepPrune/Sketch-Gated (76.5% vs. 70.6%) reflects the trade-off between pure voting and judge-based clustering, which can sometimes merge traces with different answers.

The fallback rate of 33.8% indicates that approximately one-third of traces do not find LSH candidates and trigger the full cluster scan. Despite this relatively high fallback rate, accuracy is preserved because the fallback mechanism ensures correct clustering for these traces. This demonstrates that the fallback is functioning as intended: it prevents misclustering at the cost of reduced efficiency for a subset of traces.

4.3 Ablation Study: Random-Gated Control

To understand the source of prompt reduction, we conduct an ablation study with a Random-Gated control that replaces SimHash-LSH candidate selection with uniformly random cluster selection of the same size. Table 2 shows that Random-Gated achieves nearly identical performance to Sketch-Gated: 896±1 prompts vs. 897, with identical accuracy and fallback rates.

This surprising result reveals that the prompt reduction primarily stems from the adaptive K_1 mechanism rather than the quality of LSH candidate selection. When any candidates are available (whether from LSH or random selection), the method uses $K_1^{\text{fast}} = 3$ comparisons instead of $K_1 = 10$, re-

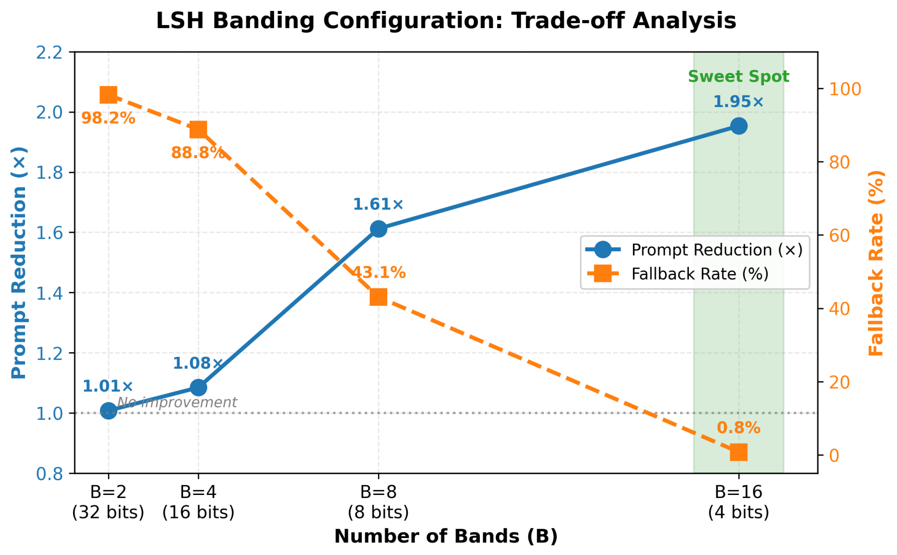

Figure 2: Effect of band count B on judge prompts and fallback rate. Increasing B reduces fallback rate exponentially while improving prompt reduction. At $B = 16$, fallback drops to 0.79% with 1.95$\times$ prompt reduction.

ducing prompts by up to 70% for those traces. The high cluster purity (>97%) for both methods indicates that even with reduced comparisons, the clustering remains accurate.

This finding has practical implications: the SimHash-LSH component provides a principled way to generate candidates, but simpler candidate generation strategies may achieve similar efficiency gains. The key insight is that the adaptive comparison budget, not the candidate quality, drives the improvement.

4.4 Banding Sensitivity Analysis

Figure 2 shows the effect of band count B on system performance. With $B = 2$ bands, the fallback rate is 98.24% and prompt reduction is minimal (1622 prompts, 1.01$\times$), as the coarse hash partitioning rarely produces matching candidates. Increasing to $B = 4$ reduces fallback to 88.77% (1507 prompts), while $B = 8$ achieves 43.07% fallback with 1014 prompts (1.61$\times$ reduction). At $B = 16$, fallback drops dramatically to 0.79% with 837 prompts (1.95$\times$ reduction), as the finer-grained banding enables more precise candidate matching.

This exponential relationship between band count and fallback rate reflects the LSH collision probability: with more bands, similar traces have higher probability of matching in at least one band, while dissimilar traces remain separated. The trade-off is computational: more bands require more hash table lookups, though this overhead is negligible compared to judge model inference. Our main experiments use $B = 8$ as a conservative choice that balances efficiency with robustness; the results suggest that $B = 16$ could further improve performance in production settings.

5 Conclusion

We presented Sketch-Gated DeepPrune, a method that accelerates inter-trace redundancy pruning by adding a lightweight SimHash-LSH filtering layer before expensive judge comparisons. On AIME and GPQA benchmarks, our approach achieves 1.82$\times$ judge prompt reduction while maintaining identical accuracy to DeepPrune. Surprisingly, our ablation study reveals that the efficiency gains primarily stem from the adaptive comparison budget mechanism rather than the quality of LSH candidate selection, suggesting that simpler gating strategies may achieve similar benefits.

Our evaluation is limited to a small-scale setting (17 problems, single model), and the finding that random gating matches sketch-based gating warrants further investigation at larger scales where semantic filtering may become more important. Future work could explore learned gating mechanisms, integration with process reward models, and application to other test-time scaling methods.

REFERENCES

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher R'e, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *ArXiv*, abs/2407.21787, 2024.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Jun-Mei Song, Ruoyu Zhang, R. Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiaoling Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, A. Liu, Bing Xue, Bing-Li Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, C. Deng, Chenyu Zhang, C. Ruan, Damai Dai, Deli Chen, Dong-Li Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, JingChang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. Cai, J. Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, K. Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, M. Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shao-Kang Wu, Tao Yun, Tian Pei, T. Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, W. Liang, Wenjun Gao, Wen-Xia Yu, Wentao Zhang, W. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, X. Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyu Jin, Xi-Cheng Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yi Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Y. Ou, Yuduan Wang, Yue Gong, Yu-Jing Zou, Yujia He, Yunfan Xiong, Yu-Wei Luo, Yu mei You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yao Li, Yi Zheng, Yuchen Zhu, Yunxiang Ma, Ying Tang, Y. Zha, Yuting Yan, Z. Ren, Z. Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhen guo Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zi-An Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645:633 – 638, 2025.

Yixin Ji, Juntao Li, Hai Ye, Kaixin Wu, Jia Xu, Linjian Mo, and Min Zhang. A survey of test-time compute: From intuitive inference to deliberate reasoning. 2025.

Joongho Kim, Xirui Huang, Zarreen Reza, Gabriel Grand, Kevin Zhu, and Ryan Lagasse. Chopping trees: Semantic similarity based dynamic pruning for tree-of-thought reasoning. *ArXiv*, abs/2511.08595, 2025.

Zhenwen Liang, Ye Liu, Tong Niu, Xiangliang Zhang, Yingbo Zhou, and Semih Yavuz. Improving llm reasoning through scaling inference computation with collaborative verification. *ArXiv*, abs/2410.05318, 2024.

Zhixiang Liang, Beichen Huang, Zheng Wang, and Minjia Zhang. Hidden states as early signals: Step-level trace evaluation and pruning for efficient test-time scaling. 2026.

H. Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, I. Sutskever, and K. Cobbe. Let's verify step by step. *ArXiv*, abs/2305.20050, 2023.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof qa benchmark. *ArXiv*, abs/2311.12022, 2023.

C. Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *ArXiv*, abs/2408.03314, 2024.

Yung-Chen Tang, Pin-Yu Chen, and Andrea Cavallaro. Carbon: Calibrated best-of-n sampling improves test-time reasoning. *ArXiv*, abs/2510.15674, 2025.

Shangqing Tu, Yaxuan Li, Yushi Bai, Lei Hou, and Juanzi Li. Deepprune: Parallel scaling without inter-trace redundancy, 2025. URL `https://arxiv.org/abs/2510.08483`.

Peiyi Wang, Lei Li, Zhihong Shao, R. Xu, Damai Dai, Yifei Li, Deli Chen, Y.Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *ArXiv*, abs/2312.08935, 2023.

Xuezhi Wang, Jason Wei, D. Schuurmans, Quoc Le, Ed H. Chi, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *ArXiv*, abs/2203.11171, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, T. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *ArXiv*, abs/2305.10601, 2023.

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

Di Zhang, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b. *ArXiv*, abs/2406.07394, 2024b.