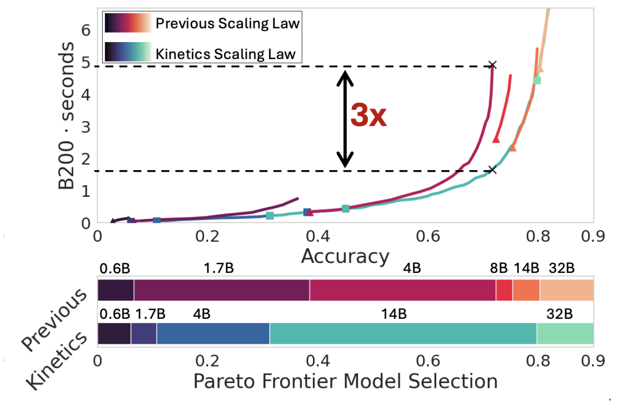
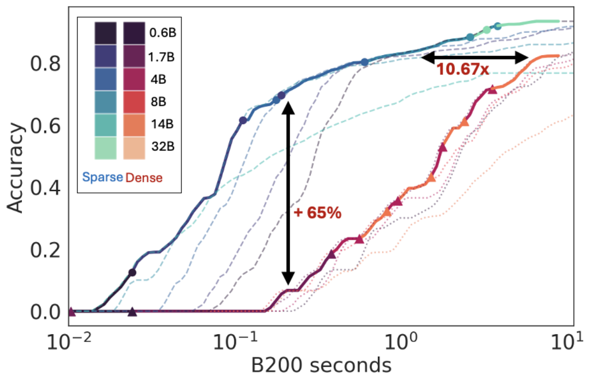
Kinetics: Rethinking Test-Time Scaling Laws

Ranajoy Sadhukhan*, Zhuoming Chen*, Haizhong Zheng, Yang Zhou, Emma Strubell, Beidi Chen*
Carnegie Mellon University
Pittsburgh, PA, USA
{rsadhukh,zhuominc,haizhonz,yangzho6,estrubel,beidic}@andrew.cmu.edu



(a) Kinetics Scaling Law



(b) Kinetics Sparse Scaling

Figure 1: (a) Pareto Frontier for Qwen3 series on AIME24. Previous test-time scaling laws [3, 64, 76] focus solely on compute optimality, neglecting the significant bottleneck of memory access in long-sequence generation. This leads to suboptimal resource utilization. By incorporating memory access, the *Kinetics Scaling Law* reduces resource demands by up to 3× to achieve the same accuracy. (b) Inspired by the Kinetics Scaling Law, we show that *sparse attention models* scale significantly better than dense models, achieving over 50-point improvements in AIME24 in the low-cost regime and consistently outperforming dense models in the high-cost regime, in addition to substantial efficiency gains. B200 second represents the amount of work performed by a single B200 at full utilization for one second.

Abstract

We rethink test-time scaling laws from a *practical efficiency* perspective, revealing that the effectiveness of smaller models is significantly overestimated. Prior work, grounded in compute-optimality, overlooks critical memory access bottlenecks introduced by inference-time strategies (e.g., Best-of-N, long CoTs). Our holistic analysis, spanning models from 0.6B to 32B parameters, reveals a new *Kinetics Scaling Law* that better guides resource allocation by incorporating both computation and memory access costs. *Kinetics Scaling Law* suggests that test-time compute is more effective when used on models above a threshold than smaller ones. A key reason is that in TTS, attention, rather than parameter count, emerges as the dominant cost factor. Motivated by this, we propose a new scaling paradigm centered on *sparse attention*, which lowers per-token cost and enables longer generations and more parallel samples within the same resource budget. Empirically, we show that sparse attention models consistently outperform dense counterparts, achieving over **60 points** gains in low-cost regimes and over **5 points** gains in high-cost regimes for problem-solving accuracy on *AIME*, encompassing evaluations on state-of-the-art MoEs.. These results suggest that sparse attention is essential for realizing the full potential of test-time scaling because, unlike training, where parameter scaling saturates, test-time accuracy continues to improve through increased generation.

[*] Both authors contributed equally to this research.

CCS Concepts

• **Computing methodologies → Machine learning**.

Keywords

Large Language Model, Scaling Law, Sparse Attention

ACM Reference Format:
Ranajoy Sadhukhan*, Zhuoming Chen*, Haizhong Zheng, Yang Zhou, Emma Strubell, Beidi Chen. 2025. Kinetics: Rethinking Test-Time Scaling Laws. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (KDD 2025 Workshop on Inference Optimization for GenAI)*. ACM, New York, NY, USA, 20 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Test-time scaling (TTS) has recently emerged as a powerful strategy (e.g., Best-of-N, Long-CoT [75]) for enhancing the reasoning capabilities of large language models (LLMs) [27, 33, 72], particularly in scenarios where agents interact with complex environments, e.g., writing code, browsing the web [57, 81] or reinforcement learning (RL) with LLMs-in-the-loop [7, 18, 31]. These capabilities, however, introduce substantial inference-time costs, making it critical to understand performance scaling in this new paradigm. Existing scaling law studies [3, 64, 76] primarily focus on floating-point operations (FLOPs) while ignoring memory access costs, which are often the dominant factor in determining wall-clock latency in TTS regimes. As shown in Figure 1a, this gap can lead to sub-optimal deployment decisions.

In Section 3, we introduce the Kinetics Scaling Law for TTS, derived from a cost model that explicitly incorporates memory access costs. This new perspective reveals markedly different conclusions about Pareto-optimal strategies for allocating test-time compute (Figure 1a). Specifically, we find that: (1) prior scaling laws consistently **overestimate** the effectiveness of small models enhanced with inference-time strategies; and (2) computational resources are best spent first on increasing model size - up to a critical threshold (empirically around 14B parameters) - before investing in test-time strategies, such as Best-of-N sampling or Long-CoTs. Guided by the Kinetics Scaling Law, our approach yields up to a **3**× resource demands reduction to reach the same accuracy on NVIDIA B200 hardware.

Our roofline analysis across a suite of state-of-the-art reasoning models reveals that the shift in optimal test-time compute strategies arises because test-time strategies (e.g., Best-of-N, Long-CoTs) disproportionately increase attention costs rather than parameter costs (Figure 2a). Our Iso-cost analysis shows that the quadratic growth of attention with generation length, combined with the disproportionate scaling of KV memory relative to model parameters, drives a preference for scaling up model size over generations. This imbalance is further exacerbated by MoE architectures [1, 12, 19, 20, 35, 62], which reduce active parameter count without alleviating attention overhead.

Building on this analysis, in Section 4 we introduce a new scaling paradigm, centered on **sparse** attention, which fundamentally reshapes the scaling law and significantly enhances the scalability of TTS (Figure 1b). According to our Kinetics Sparse Scaling Law, computational resources are best allocated to test-time strategies rather than reducing sparsity. As more computing is invested at test time, higher sparsity becomes increasingly critical to fully leveraging the benefits of these strategies. Guided by this principle, it increases problem-solving rates by up to **60** points in the low-cost regime and over **5** points in the high-cost regime on AIME24 and LiveCodeBench, encompassing evaluations on state-of-the-art MoEs.through massive generated tokens, which is unaffordable for dense counterparts.

In Section 5, we demonstrate the practicality of the Kinetics Sparse Scaling Law using a simple block-sparse attention mechanism built on top of paged attention. This approach achieves up to a **25**× wall-clock speedup on H200 GPUs. While sparsity has traditionally been employed either for regularization in small models [55, 73] or to reduce computation in over-parameterized networks [5, 14, 21, 29, 46, 53], our work introduces a fundamentally different perspective: **sparsity as a central enabler of efficient and scalable test-time inference**. In contrast to pretraining – where scaling laws often exhibit diminishing returns [32] – TTS continues to benefit from increased token generation and more optimized inference paths. We hope this study can guide and encourage future co-design of model architectures, test-time strategies, and hardware systems to fully unlock the next wave of scaling at deployment.

2 Cost Model and eFLOPs

We propose a cost model that captures both compute and memory access overhead during inference, focusing on realistic deployment settings (batch size ≫ 1, model parallelism, and shared prompt cache). Notation is in Table 1.

Computation and Memory. Following [3], the compute cost combines linear layer operations and self-attention:

$$C_{\text{comp}} = 2PL_{\text{out}} + r(2L_{\text{in}} + L_{\text{out}})L_{\text{out}}D$$

Memory access includes both parameter loading and KV cache reads:

$$C_{\text{mem}} = 2PL_{\text{out}} + 2L_{\text{in}}L_{\text{out}}D + L_{\text{out}}^2D$$

In practice, parameter loading is amortized across large batches [16], so we omit that term and share prompt KV cache across N trials. The final per-task compute and memory cost becomes:

$$C_{\text{comp}}(N) = 2PNL_{\text{out}} + 2rNL_{\text{in}}L_{\text{out}}D + rNL_{\text{out}}^2D \quad (1)$$

$$C_{\text{mem}}(N) = 2L_{\text{in}}L_{\text{out}}D + NL_{\text{out}}^2D \quad (2)$$

eFLOPs. We define eFLOPs (equivalent FLOPs) as a linear combination of compute and memory cost, scaled by hardware intensity I to capture the memory and computational operations under the same scale:

$$\text{eFLOPs} = C_{\text{comp}} + C_{\text{mem}} \cdot I$$

Analysis. Our key insight is **attention-related cost dominates in long CoTs.** We show this by estimating the ratio of attention-related cost to parameter-related cost Φ.

$$\Phi = \frac{2rL_{in}D + (rD + ID)L_{out}}{2P}$$

As shown in Figure 2a, in the regime of long CoTs, where the generation length exceeds 4096 tokens, the cost of attention surpasses that of model parameters by a factor of 100 ∼ 1000. **MLA** [45] reduces KV memory access by a constant factor (similar to r in GQA), it is insufficient for achieving true scalability due to several limitations: (1) MLA does not reduce attention computation; (2) the gap between FLOPs and memory bandwidth is expected to widen in the future; and (3) emerging **fine-grained MoEs** [1, 12, 65] drastically reduce FLOPs in linear layers by a factor of 10 ∼ 20×, further increasing the relative cost of attention.

Under the context of Long-CoTs being widely adopted, we can safely assume generated length $L_{out} \gg L_{in}$ or *at least* proportional to L_{in}. Hence, the bottleneck of inference is shifted from linear term $L_{out}P$ to the quadratic term L_{out}^2D, motivating our Kinetics Scaling Law, akin to kinetic energy: $E_k = \frac{1}{2}mv^2$.

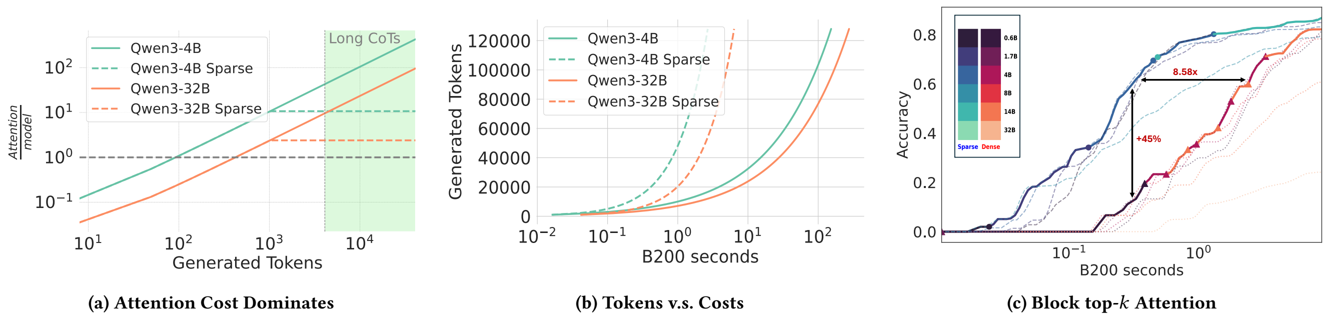


Figure 2: (a) Inference cost is dominated by attention, which is $100 \sim 1000\times$ more than model parameter computation, sparse attention fundamentally mitigates this bottleneck. (b) Under the same resource constraint, sparse attention can generate massive tokens out of the reach of dense models, which is proven to enhance the effectiveness of test-time scaling. (c) Simple block sparse attention yields substantial gains—improving accuracy by 45 points in the low-cost regime and achieving equivalent accuracy while using $8.58\times$ fewer resources.

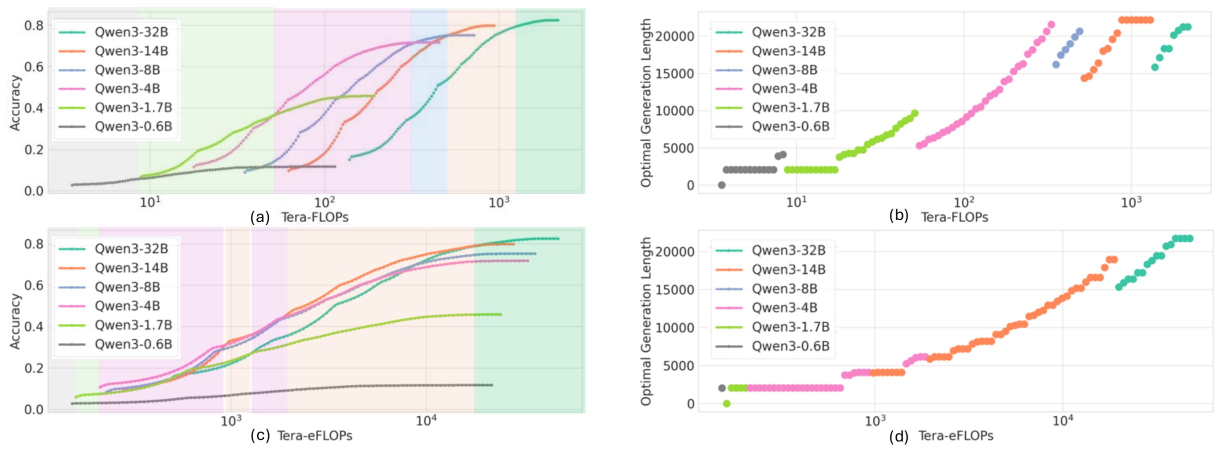


Figure 3: AIME Pareto Frontier (Long-CoTs). We launch evaluations for Qwen3 series models. We control the inference cost in eFLOPs (ab for our scaling law) or FLOPs (cd for previous scaling law) and measure the accuracy in AIME24. The optimal model is marked with different colors in (ac). The optimal generation length is in (bd).

More details are in Appendix A.

2.1 Experimental Setup

Experimental Setup. Tasks. we focus on three challenging reasoning benchmarks: AIME24 [50], AIME25 [51], math datasets spanning algebra, combinatorics, and geometry, and LiveCodeBench [34][1], which includes complex programming problems from recent coding competitions. **Models.** We evaluate performance across various model sizes of the Qwen3 [78] and DeepSeek-R1-Distilled-Qwen [27, 79] series. **Test-time Strategies.** To eliminate the confounding effects introduced by the specific implementations of test-time strategies—such as the quality of reward models, we adopt two representative yet straightforward approaches: Long-CoTs, a practical and widely used method in state-of-the-art reasoning models, and the *oracle* Best-of-N (repeated sampling [3]), which measures the solving rate for verifiable problems and suggests an upper bound via TTS. **Hardware.** We use the specifications of NVIDIA B200 as hardware reference to study the latest serving scenarios. Experiments details are presented in Appendix D.

[1]For LiveCodeBench, we sample 50 problems from the *v5* subset (24 hard, 16 medium, 10 easy).

3 Rethinking Test-time Scaling Law

we study the scaling behavior of Qwen3 [78]. In the *Long CoTs* setting (single trial per question, $N_T = 1$), we vary generation length n_T to evaluate performance across cost levels. Results in Figure 3 reveal two key findings of our Kinetics Scaling Law.

- **Efficiency of small models is overestimated.** As shown in Figure 3 **(a, c)**, smaller models like 4B and 8B are outperformed by the 14B model even at low accuracy levels (e.g., below 40%). The 0.6B model appears on the Pareto frontier only when accuracy is negligible. In contrast to prior scaling laws, which gave smaller models more prominence, our results show they are often suboptimal in practice.

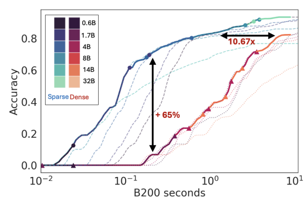
- **CoT length more effective than parameter size only beyond a critical model scale (empirically, 14B).** The Kinetics Scaling Law shows that, under limited compute, scaling up the model yields greater benefits than extending CoT length. As seen in Figure 3 **(b, d)**, only the 14B and 32B models gain from CoTs longer than 10K tokens. For smaller models (e.g., 1.7B and 4B), switching to a larger model is more effective when $L_{\text{out}} < 5\text{K}$. This suggests compute should primarily be allocated to increasing model size, not generation length (Figure 3 **(d)**). In contrast, previous scaling laws assumed longer CoTs consistently improved performance across all model sizes and only favored model scaling once those gains plateaued.

More details are in Appendix B.

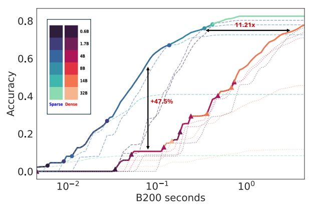
4 Sparse Test-time Scaling Law

Based on our findings in Section 3, we propose a new scaling paradigm centered on sparse attention. Sparse attention fundamentally reshapes the Kinetics Scaling Law in Section 3 and enhances the scalability of TTS.

Sparse attention significantly enhances problem-solving performance. As shown in Figures 4a and 4b, compared to dense baselines, for both of the inference strategies and models of various sizes, sparse attention models improve problem-solving rates by up to 60 points in the low-cost regime and over 5 points in the high-cost regime. From an efficiency perspective, dense models require over 10× more eFLOPs to match the same solving rate. These findings underscore sparse attention as a key enabler for unlocking the full benefits of test-time scaling.



(a) ***Best-of-N*** **Scaling**



(b) ***Long-CoTs*** **Scaling**

Figure 4: Sparse Attention Boosts Test-Time Scaling. We show that sparse attention models significantly improve the cost-accuracy trade-off under both inference strategies.

Sparse attention becomes increasingly valuable in high-cost scenarios. We investigate the tradeoff between KV budget B and reasoning trials (N). Our analysis reveals a consistent trend: allocating additional compute toward generating more tokens is generally more effective than expanding the KV cache. In *Best-of-N* frontier, doubling the cost leads to only a 1.18× increase in KV budget, compared to a 1.74× increase in total generated tokens. (Figures 20a to 20d)

Sparse attention reshapes the Kinetics Scaling Law. As shown in Figure 5, applying sparse attention significantly improves the efficiency of smaller models (0.6B, 1.7B, 4B), allowing them to re-emerge on the Pareto frontier across a broader range. Sparse attention reduces attention cost from a quadratic cost term (L^2D) to a linear one (LBD), making it comparable when compared to the cost of computing with model parameters (LP).

More details are in Appendices C and D.

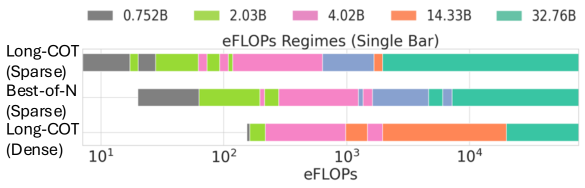


Figure 5: Compared to the scaling law for the dense models, small models are more effective with sparse attention. They occupy more space in the Pareto Frontier.

5 Experimental Validation

Top-k attention is theoretically appealing but impractical. We adopt block top-k attention as a tractable alternative for two reasons: it exploits temporal locality to retrieve relevant KV blocks [66], and it integrates efficiently with hardware-friendly paged attention [39]. Each block is scored via averaged key vectors, and importance scores are shared across heads via GQA. The implementation details are provided in Appendix D.2. We evaluate task throughput—the number of correct tasks completed per second—as a measure of test-time efficiency. On 8×H200 GPUs with batch size 4096, block top-k significantly improves throughput across models. For Qwen3-0.6B, throughput improves 12.6× to 25× from 16k to 32k tokens (Figure 6). Sparse attention mitigates dense attention's inefficiencies, restoring small model utility under compute constraints.

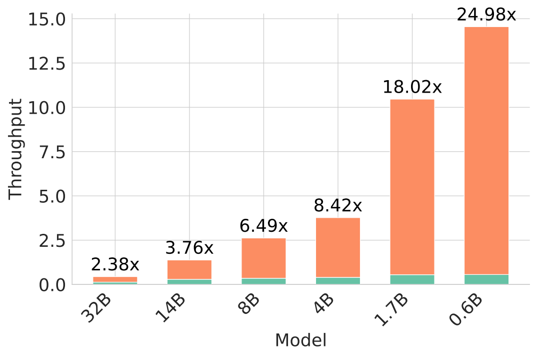


Figure 6: Task throughput with block top-k attention.

6 Conclusion and Discussion

This work introduces the *Kinetics Scaling Law* based on the insight that attention costs rather than parameter counts are the dominant factor in TTS. We demonstrate that sparse attention fundamentally reshapes the scaling landscape, enabling longer generations and significantly higher accuracy. We envision the Kinetics Scaling Law as a foundational tool for guiding end-to-end design across LLM serving, agent frameworks, and reinforcement learning environments. Kinetics Sparse Scaling may signal a new paradigm, enabling continued progress even beyond pretraining plateaus. While our analysis centers on NVIDIA GPUs, the underlying principle that scaling memory bandwidth is more challenging and costly than scaling FLOPs applies broadly across hardware platforms. Ultimately, this study highlights the need for co-designing model architectures, test-time inference techniques, and hardware infrastructure as a critical step toward enabling the next wave of scalable LLM deployment.

References

[1] AI@Meta. 2025. Llama 4 Model Card. (2025). https://github.com/meta-llama/llama-models/blob/main/models/llama4/MODEL_CARD.md

[2] Daman Arora and Andrea Zanette. [n. d.]. Training language models to reason efficiently, 2025. *URL https://arxiv. org/abs/2502.04463* ([n. d.]).

[3] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787* (2024).

[4] Ruisi Cai, Yuandong Tian, Zhangyang Wang, and Beidi Chen. 2024. Lococo: Dropping in convolutions for long context compression. *arXiv preprint arXiv:2406.05317* (2024).

[5] Beidi Chen, Tri Dao, Eric Winsor, Zhao Song, Atri Rudra, and Christopher Ré. 2021. Scatterbrain: Unifying sparse and low-rank attention. *Advances in Neural Information Processing Systems* 34 (2021), 17413–17426.

[6] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318* (2023).

[7] Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. 2025. Reinforcement Learning for Long-Horizon Interactive LLM Agents. *arXiv preprint arXiv:2502.01600* (2025).

[8] Mouxiang Chen, Binyuan Hui, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Jianling Sun, Junyang Lin, and Zhongxin Liu. 2025. Parallel Scaling Law for Language Models. *arXiv preprint arXiv:2505.10475* (2025). arXiv:2505.10475 [cs.LG] https://arxiv.org/abs/2505.10475

[9] Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, et al. 2024. Magicpig: Lsh sampling for efficient llm generation. *arXiv preprint arXiv:2410.16179* (2024).

[10] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509* (2019).

[11] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. 2020. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794* (2020).

[12] Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066* (2024).

[13] Tri Dao. 2023. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. *CoRR* abs/2307.08691 (2023). doi:10.48550/ARXIV.2307.08691 arXiv:2307.08691

[14] Tri Dao, Beidi Chen, Kaizhao Liang, Jiaming Yang, Zhao Song, Atri Rudra, and Christopher Ré. 2021. Pixelated Butterfly: Simple and Efficient Sparse Training for Neural Network Models. In *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/2112.00029

[15] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (Eds.).

[16] DeepSeek-AI. 2025. DeepSeek Open Infra Index. (2025). https://github.com/deepseek-ai/open-infra-index/blob/main/202502OpenSourceWeek/day_6_one_more_thing_deepseekV3R1_inference_system_overview.md

[17] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems* 35 (2022), 30318–30332.

[18] Danny Driess, Minh Nguyen, Fei Xia, et al. 2023. PaLM-E: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378* (2023).

[19] Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Dehao Chen, Yonghui Wu, and Jeff Dean. 2021. GLaM: Efficient Scaling of Language Models with Mixture-of-Experts. *arXiv preprint arXiv:2112.06905* (2021).

[20] William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *Journal of Machine Learning Research* 23, 1 (2022), 5232–5270.

[21] Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*. PMLR, 10323–10337.

[22] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323* (2022).

[23] Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Aurick Qiao, and Hao Zhang. 2024. Efficiently Serving LLM Reasoning Programs with Certaindex. *arXiv preprint arXiv:2412.20993* (2024).

[24] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).

[25] Albert Gu and Tri Dao. 2023. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *CoRR* abs/2312.00752 (2023). doi:10.48550/ARXIV.2312.00752 arXiv:2312.00752

[26] Albert Gu, Karan Goel, and Christopher Ré. 2022. Efficiently Modeling Long Sequences with Structured State Spaces. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net. https://openreview.net/forum?id=uYLFoz1vlAC

[27] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* (2025).

[28] Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2024. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769* (2024).

[29] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. 2021. Sparsity in Deep Learning: Pruning and Growth for Efficient Inference and Training in Neural Networks. *Journal of Machine Learning Research* 22, 241 (2021), 1–124.

[30] Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. 2024. KVQuant: Towards 10 Million Context Length LLM Inference with KV Cache Quantization. In *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (Eds.), Vol. 37. Curran Associates, Inc., 1270–1303. https://proceedings.neurips.cc/paper_files/paper/2024/file/028fcbcf85435d39a40c4d61b42c99a4-Paper-Conference.pdf

[31] Wenlong Huang, Fei Fei, and Chelsea Finn. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *arXiv preprint arXiv:2201.07207* (2022).

[32] Sutskever Ilya. [n. d.]. Ilya Sutskever: "Sequence to sequence learning with neural networks: what a decade". https://www.youtube.com/watch?v=1yvBqasHLZs

[33] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720* (2024).

[34] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code. *arXiv preprint arXiv:2403.07974* (2024).

[35] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088* (2024).

[36] Jordan Juravsky, Bradley Brown, Ryan Ehrlich, Daniel Y Fu, Christopher Ré, and Azalia Mirhoseini. 2024. Hydragen: High-throughput llm inference with shared prefixes. *arXiv preprint arXiv:2402.05099* (2024).

[37] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 5156–5165. http://proceedings.mlr.press/v119/katharopoulos20a.html

[38] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer. In *The International Conference on Machine Learning (ICML)*.

[39] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. arXiv:2309.06180 [cs.LG] https://arxiv.org/abs/2309.06180

[40] Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*. PMLR, 19274–19286.

[41] Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. SnapKV: LLM Knows What You are Looking for Before Generation. arXiv:2404.14469 [cs.CL] https://arxiv.org/abs/2404.14469

[42] Chaofan Lin, Jiaming Tang, Shuo Yang, Hanshuo Wang, Tian Tang, Boyu Tian, Ion Stoica, Song Han, and Mingyu Gao. 2025. Twilight: Adaptive Attention Sparsity with Hierarchical Top-p Pruning. *arXiv preprint arXiv:2502.02770* (2025).

[43] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems* 6 (2024), 87–100.

[44] Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. 2024. Qserve: W4a8kv4 quantization and system co-design for efficient llm serving. *arXiv preprint arXiv:2405.04532* (2024).

[45] Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434* (2024).

[46] Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. 2023. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*. PMLR, 22137–22176.

[47] Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. 2024. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750* (2024).

[48] Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. 2025. Reasoning Models Can Be Effective Without Thinking. *arXiv preprint arXiv:2504.09858* (2025).

[49] Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025. CoT-Valve: Length-Compressible Chain-of-Thought Tuning. *arXiv preprint arXiv:2502.09601* (2025).

[50] MAA. 2024. American Invitational Mathematics Examination 2024. https://artofproblemsolving.com/wiki/index.php/American_Invitational_Mathematics_Examination?srsltid=AfmBOoqiDCiaGTLQrsRTKsZui8RFnjOZqM4qIqY3yGB3sBaqOaxwf_Xt

[51] MAA. 2025. American Invitational Mathematics Examination 2025. https://artofproblemsolving.com/wiki/index.php/American_Invitational_Mathematics_Examination?srsltid=AfmBOoqiDCiaGTLQrsRTKsZui8RFnjOZqM4qIqY3yGB3sBaqOaxwf_Xt

[52] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, et al. 2023. Specinfer: Accelerating generative large language model serving with tree-based speculative inference and verification. *arXiv preprint arXiv:2305.09781* (2023).

[53] Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. 2021. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378* (2021).

[54] Amirkeivan Mohtashami, Matteo Pagliardini, and Martin Jaggi. 2023. CoTFormer: A Chain-of-Thought Driven Architecture with Budget-Adaptive Computation Cost at Inference. *arXiv preprint arXiv:2310.10845* (2023).

[55] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. 2017. Variational Dropout Sparsifies Deep Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2498–2507.

[56] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393* (2025).

[57] Reiichiro Nakano, Jacob Hilton, Jeffrey Wu, et al. 2021. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332* (2021).

[58] Piotr Nawrot, Robert Li, Renjie Huang, Sebastian Ruder, Kelly Marchisio, and Edoardo M Ponti. 2025. The Sparse Frontier: Sparse Attention Trade-offs in Transformer LLMs. *arXiv preprint arXiv:2504.17768* (2025).

[59] Xuefei Ning, Zinan Lin, Zixuan Zhou, Zifu Wang, Huazhong Yang, and Yu Wang. 2023. Skeleton-of-thought: Prompting llms for efficient parallel generation. *arXiv preprint arXiv:2307.15337* (2023).

[60] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems* 5 (2023), 606–624.

[61] Ranajoy Sadhukhan, Jian Chen, Zhuoming Chen, Vashisth Tiwari, Ruihang Lai, Jinyuan Shi, Ian En-Hsu Yen, Avner May, Tianqi Chen, and Beidi Chen. 2024. Magicdec: Breaking the latency-throughput tradeoff for long context generation with speculative decoding. *arXiv preprint arXiv:2408.11049* (2024).

[62] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. *arXiv preprint arXiv:1701.06538* (2017).

[63] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. 2023. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*. PMLR, 31094–31116.

[64] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314* (2024).

[65] Snowflake-Team. 2024. Snowflake Arctic. https://github.com/Snowflake-Labs/snowflake-arctic. Apache 2.0 License.

[66] Hanshi Sun, Zhuoming Chen, Xinyu Yang, Yuandong Tian, and Beidi Chen. 2024. TriForce: Lossless Acceleration of Long Sequence Generation with Hierarchical Speculative Decoding. arXiv:2404.11912 [cs.CL] https://arxiv.org/abs/2404.11912

[67] Hanshi Sun, Momin Haider, Ruiqi Zhang, Huitao Yang, Jiahao Qiu, Ming Yin, Mengdi Wang, Peter Bartlett, and Andrea Zanette. 2024. Fast best-of-n decoding via speculative rejection. *arXiv preprint arXiv:2410.20290* (2024).

[68] Ruslan Svirschevski, Avner May, Zhuoming Chen, Beidi Chen, Zhihao Jia, and Max Ryabinin. 2024. Specexec: Massively parallel speculative decoding for interactive llm inference on consumer devices. *Advances in Neural Information Processing Systems* 37 (2024), 16342–16368.

[69] Jihoon Tack, Jack Lanchantin, Jane Yu, Andrew Cohen, Ilia Kulikov, Janice Lan, Shibo Hao, Yuandong Tian, Jason Weston, and Xian Li. 2025. LLM Pretraining with Continuous Concepts. *arXiv preprint arXiv:2502.08524* (2025).

[70] Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. 2024. Quest: Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774* (2024).

[71] NovaSky Team. 2025. Sky-T1: Train your own O1 preview model within $450. https://novasky-ai.github.io/posts/sky-t1. Accessed: 2025-01-09.

[72] Qwen Team. 2025. QwQ-32B: Embracing the Power of Reinforcement Learning. https://qwenlm.github.io/blog/qwq-32b/

[73] Robert Tibshirani. 1996. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)* 58, 1 (1996), 267–288.

[74] Ajay Tirumala and Raymond Wong. 2024. Nvidia blackwell platform: Advancing generative ai and accelerated computing. In *2024 IEEE Hot Chips 36 Symposium (HCS)*. IEEE Computer Society, 1–33.

[75] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems* 35 (2022), 24824–24837.

[76] Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2024. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724* (2024).

[77] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient Streaming Language Models with Attention Sinks. arXiv:2309.17453 [cs.CL] https://arxiv.org/abs/2309.17453

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

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

[80] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems* 36 (2023), 11809–11822.

[81] Shinn Yao, Jiaming Zhao, Dian Yu, et al. 2023. ReAct: Synergizing reasoning and acting in language models. *Advances in Neural Information Processing Systems (NeurIPS)* (2023).

[82] Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yineng Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod Grover, Arvind Krishnamurthy, and Luis Ceze. 2025. FlashInfer: Efficient and Customizable Attention Engine for LLM Inference Serving. *arXiv preprint arXiv:2501.01005* (2025). https://arxiv.org/abs/2501.01005

[83] Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. 2022. Orca: A Distributed Serving System for Transformer-Based Generative Models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*. USENIX Association, Carlsbad, CA, 521–538. https://www.usenix.org/conference/osdi22/presentation/yu

[84] Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, YX Wei, Lean Wang, Zhiping Xiao, et al. 2025. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089* (2025).

[85] Zhihang Yuan, Yuzhang Shang, Yang Zhou, Zhen Dong, Zhe Zhou, Chenhao Xue, Bingzhe Wu, Zhikai Li, Qingyi Gu, Yong Jae Lee, et al. 2024. Llm inference unveiled: Survey and roofline model insights. *arXiv preprint arXiv:2402.16363* (2024).

[86] Amir Zandieh, Insu Han, Majid Daliri, and Amin Karbasi. 2023. Kdeformer: Accelerating transformers via kernel density estimation. In *International Conference on Machine Learning*. PMLR, 40605–40623.

[87] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*

(2022).

[88] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems* 36 (2023), 34661–34710.

[89] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. 2024. Sglang: Efficient execution of structured language model programs. *Advances in Neural Information Processing Systems* 37 (2024), 62557–62583.

Table of Contents

Table 1: Notation Used throughout the Paper.

Symbol	Description	Symbol	Description
$T, \mathcal{T}$	Task (set)	L_{out}	# Gen tokens
M	Model	N, N_T	Reasoning trials
$C, C_{\text{TTS}}(\cdot)$	Cost function	n, n_T	Max # tokens
$\mathcal{A}$	Algorithm	B, B_T	KV budget
L_{in}	Prompt length	P	Parameters
D	KV size / token	r	GQA ratio

A Cost Model

In this section, we delve into the cost models used in the Kinetics Scaling Law. We show empirically that adopting a max cost model does not alter the scaling behavior and outline methods for calculating the cost of sparse attention models. Notation is in Table 1.

A.1 Full Formulations of Cost Model

We first calculate the inference cost for the cases where the batch size is 1, and then extend to a more general case in TTS. Finally, we propose our cost model using equivalent FLOPs.

Computation. As discussed in [3], the computation consists of two parts: linear modules and self-attention, which is (we assume the model is served in BFloat16.)

$$C_{\text{comp}} = \underbrace{2PL_{out}}_{\text{model parameters computation}} + \underbrace{r(2L_{in} + L_{out})L_{out}D}_{\text{self-attention}}$$

Memory Access. Memory access also consists of two parts: model parameters and KV cache.

$$C_{\text{mem}} = \underbrace{2PL_{out}}_{\text{model parameter access}} + \underbrace{2L_{in}L_{out}D}_{\text{prompt KV cache}} + \underbrace{L_{out}^2 D}_{\text{decoding KV cache}}$$

In real serving scenarios, a large batch size will be used [16] with growing GPU VRAM [74] and model parallelism [60]. The access to the model parameter will be amortized across requests in a batch shows parameter access time is negligible when the batch size is large). Thus, we only consider the second term (i.e., KV cache loading) in our cost function. Furthermore, in the cases that we have N reasoning trials, the prompt cache access [36, 89] is also shared across these N trials. Thus,

$$C_{\text{comp}}(N) = 2PNL_{out} + 2rNL_{in}L_{out}D + rNL_{out}^2D \tag{3}$$

$$C_{\text{mem}}(N) = 2L_{in}L_{out}D + NL_{out}^2D \tag{4}$$

eFLOPs. We propose eFLOPs (equivalent FLOPs) to capture both compute and memory access cost,

$$\text{eFLOPs} = C_{\text{comp}} + C_{\text{mem}} \times I \tag{5}$$

where I is the arithmetic intensity of hardware, which reflects that modern accelerators usually have a much larger computation capacity over memory bandwidth, and the gap is growing over the years [61]. In this work, we use $I = 562.5$ (unit: FLOPs × s / GB) from NVIDIA B200 [74].

With Equations (3) to (5), we obtain the final cost model.

$$C_{\text{TTS}} = \underbrace{2NPL_{out}}_{\text{linear modules computation}} + \underbrace{2rNL_{in}DL_{out} + rNDL_{out}^2}_{\text{self-attention computation}} + \underbrace{2IL_{in}DL_{out} + INDL_{out}^2}_{\text{KV access}} \tag{6}$$

where P, r, D are hyper-parameters determined by model M[2].

A.2 Max Cost Model v.s. Additive Cost Model

Max cost model is widely used in performance modeling [85]. It assumes that computation and memory operations can be fully overlapped with each other and only considers the bottleneck operation for cost measurement.

$$C_{\text{max-cost}} = \max(C_{\text{comp}}, C_{\text{mem}} \times I)$$

where C_{comp} denotes the compute cost, C_{mem} the memory cost per access, and I the memory intensity.

In this section, we analyze the Kinetics Scaling Law using the max cost model. For clarity, we refer to the cost model $C_{\text{comp}} + C_{\text{mem}} \times I$, which is used in the main paper, as **the additive cost model**.

We draw two conclusions from empirical results **under the max cost model**:

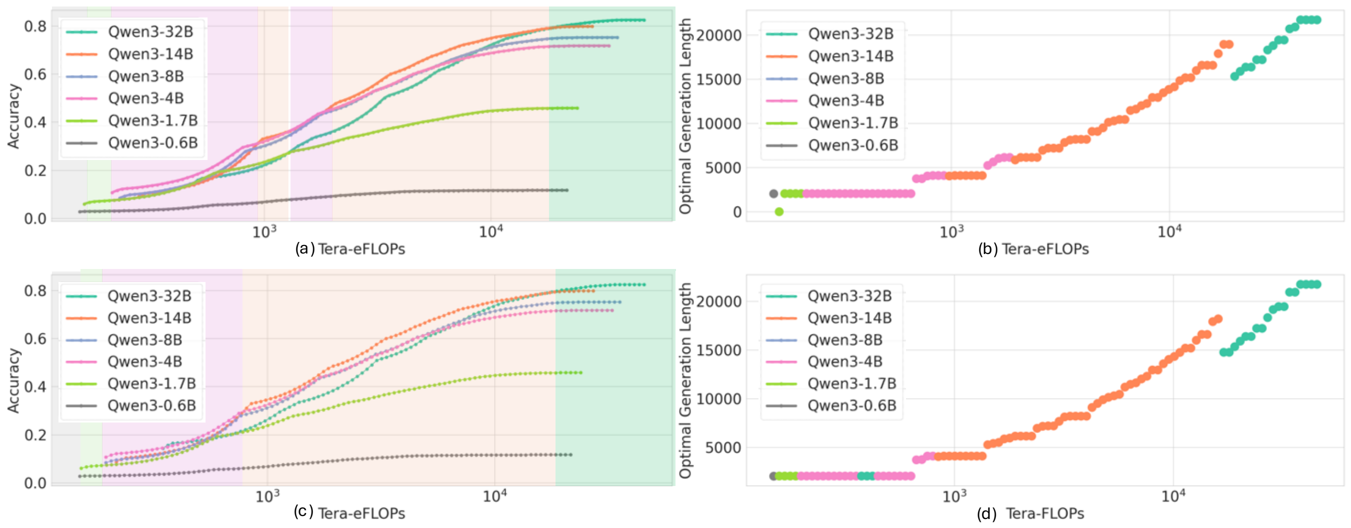


Figure 7: AIME Pareto Frontier (Long-CoTs) with Max Cost Models. (a)(b) is the original plot with the additive cost model. (c)(d) is the corresponding plot using max cost models. Compared to the original plots, the overall trend is similar except that larger models span a slightly broader region on the Pareto frontier. For example, the 14B model now consistently outperforms the 4B model with a noticeable gap around accuracy 0.3 and maintains dominance thereafter. In contrast, under the additive cost model in Figure 3(a), the two models alternate in performance until accuracy exceeds 0.4. This suggests that, when evaluated using a max cost model, larger models appear slightly more efficient relative to their performance under additive cost models.

[2]Since L_{out} might differ across reasoning trials, we take the expectation for $\mathbb{E}[L_{out}]$ and $\mathbb{E}[L_{out}^2]$.

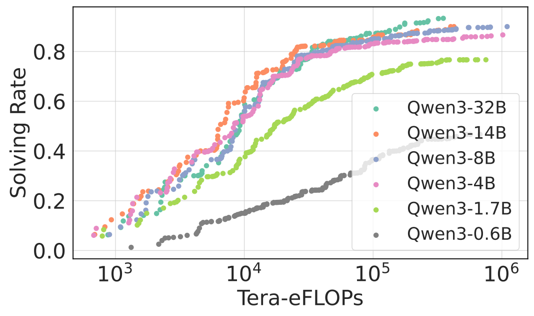
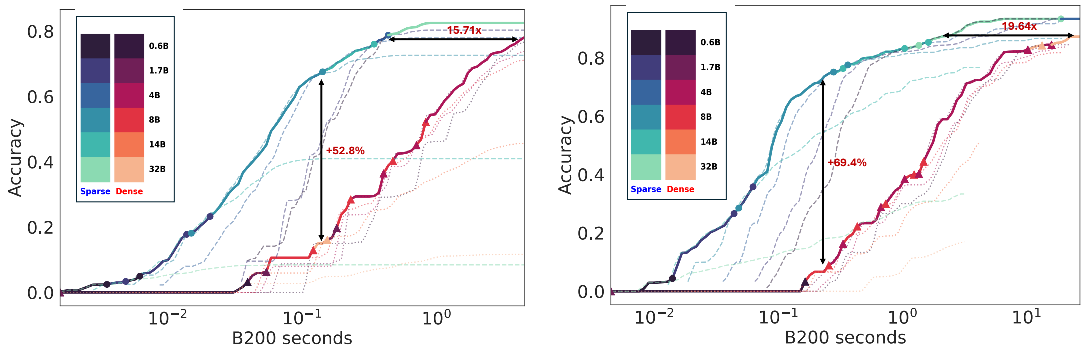


Figure 8: AIME Pareto Frontier (Best-of-N) with Max Cost Models. We re-plot Figure 10a using max cost models. The Pareto Frontier is very similar under different cost models.



(a) Long-CoTs **(b) Best-of-N**

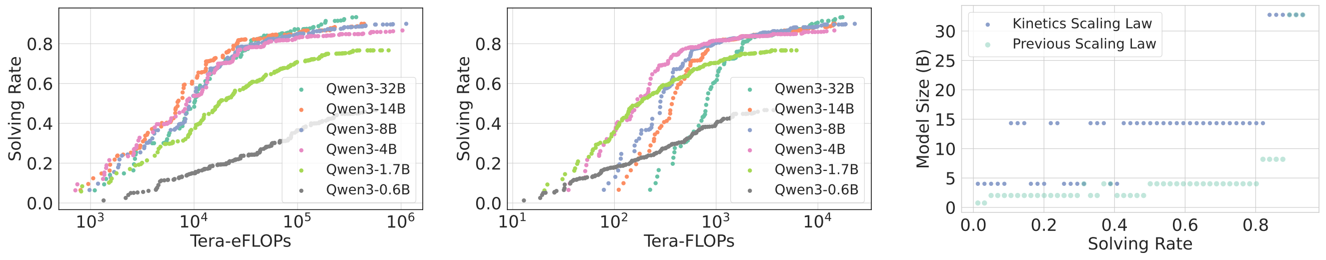
Figure 9: Sparse attention scales significantly better under max cost models. We re-plot Figures 4a and 4b using max cost models. Compared to the original plots, the performance and efficiency gaps between sparse attention models and dense models become more pronounced. In Long-CoTs, the accuracy and efficiency gaps increase from 47.5 points and 11.21× to 52.8 points and 15.71×, respectively. In Best-of-N, the gaps widen from 65 points and 10.67× to 69.4 points and 19.64×.

- **Kinetics scaling law for dense models still holds.** We re-plot Figure 3**(a)(b)** and Figure 10a under the measurement of max cost models in Figures 7 and 8. We find except that in Long-CoTs scenarios, large models become slightly more effective in low-cost regime (with accuracy~0.3), the overall trends are very close to the plots with additive cost models.
- **Sparse attention solves problems more cost-effectively.** We re-plot Figures 4a and 4b in Figures 9a and 9b. Under the max cost models, in Long-CoTs, the accuracy and efficiency gaps increase from 47.5 points and 11.21× to 52.8 points and 15.71×, respectively. In Best-of-N, the gaps widen from 65 points and 10.67× to 69.4 points and 19.64×. These results indicate that under the max cost model, our claim that sparse attention can enhance problem-solving performance is strengthen. Compared to dense attention models, sparse attention models tend to have more balanced memory and compute costs. Thus omitting one of them via a max cost model will favor sparse attention models.

A.3 Details about Sparse Attention Cost Model

Sparse attention models follow different cost functions due to the sparsification of KV memory access. In this paper, we focus on algorithms that impose a uniform KV budget (denoted as B) per attention head for each decoded token. We consider $L_{in} \geq B$ for the sake of simplicity. Under this setting, the cost model for sparse attention is given by:

$$C_{\text{sparse}} = \underbrace{2NPL_{\text{out}} + 2rNDBL_{\text{out}}}_{\text{compute}} + \underbrace{2INDBL_{\text{out}}}_{\text{memory}}. \tag{7}$$



(a) Accuracy (eFLOPs) **(b) Accuracy (FLOPs)** **(c) Optimal Models**

Figure 10: AIME24 Score Curve Envelope (Best-of-N). We control the incurred inference cost in eFLOPs (a) or FLOPs (b) and measure the solving rate (Coverage) in AIME24 for various models by varying the maximum allowed number of reasoning trials. By taking the curve envelopes, we can project the optimal models in (c).

In practical implementations, we must also account for the overhead associated with retrieving or searching KV memory, denoted as C_{search}, which depends on the specific sparse attention algorithm $\mathcal{A}$. For example, in block top-k selection, the search cost is:

$$C_{\text{search}} = \underbrace{\frac{2NL_{\text{in}}DL_{\text{out}} + rNDL_{\text{out}}^2}{2\text{Block-Size}}}_{\text{compute}} + \underbrace{\frac{2IL_{\text{in}}DL_{\text{out}} + INDL_{\text{out}}^2}{2\text{Block-Size}}}_{\text{memory}}. \tag{8}$$

In our work, we choose the Block-Size in such a way that C_{sparse} and C_{search} are roughly balanced, so that the sparse attention cost increases sub-linearly with generation length.

For local attention and oracle top-k attention, we assume no search overhead, i.e., $C_{\text{search}} = 0$.

Many sparse attention algorithms skip the first layer [9, 70, 88], resulting in only a minor increase in total cost. For the Qwen3 series, this additional overhead is bounded by 3.57% for the 0.6B model and by 1.56% for the 32B model.

B Dense Scaling Law

In this section, we further verify Kinetics Scaling Law for dense models proposed in Section 3 with Iso-Cost analysis and extended experimental results of different benchmarks and model series.

B.1 Best-of-N AIME24

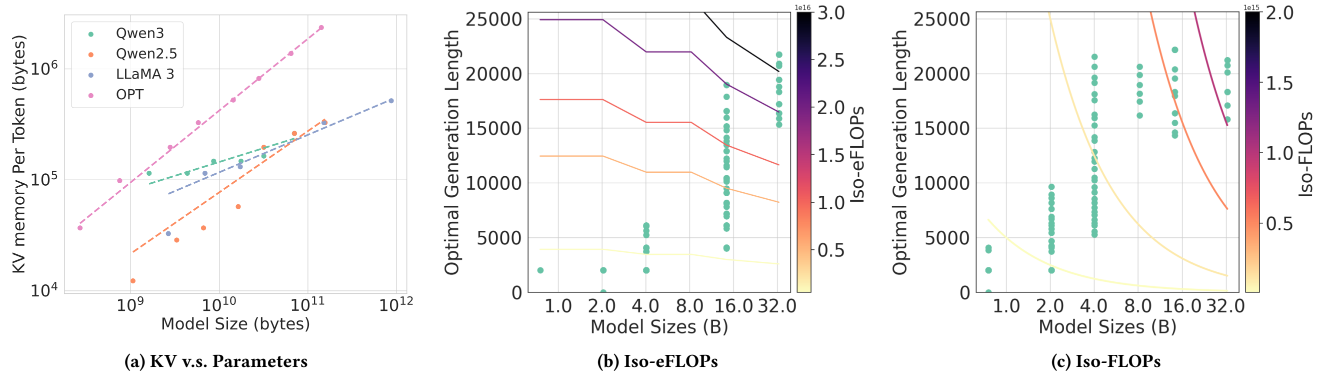
In the *Best-of-N* setting, we fix the maximum number of generated tokens at n_T, and vary the number of reasoning trials N to evaluate the problem-solving rate (i.e., the probability that at least one trial produces a correct answer). We have similar observations in Figures 10a to 10c. Under the previous scaling laws (Figure 10b), the most cost-effective strategy to achieve high accuracy is to apply repeated sampling using smaller models. Kinetics Scaling Law Figure 10a reveals that deploying a 14B model with fewer reasoning trials is more efficient. We also observe a critical size of 14B. For models smaller than 14B, increasing compute is best allocated toward model scaling rather than additional trials. For models at or above 14B, however, further computation is more effectively spent on increasing the number of reasoning trials, up to diminishing returns.

B.2 Iso-Cost Study

We attribute the above divergence between Kinetics and previous scaling laws to two reasons.

Disproportionation between KV memory size D and model parameters P. Smaller models tend to require significantly more KV cache relative to their parameter size. For example, Qwen3-0.6B demands 3.5GB of KV cache to store 32K tokens, despite the model itself occupying only 1.2GB. In contrast, Qwen3-32B uses just 8GB of KV cache for the same sequence length. Empirically, doubling model parameters results in only a 1.18× increase in KV cache size. As shown in Figure 11a, this phenomenon is consistently observed across model families such as OPT [87] (1.55×), Qwen2.5 [79] (1.46×), and LLaMA3 [24] (1.27×).

Shift from linear to quadratic cost model. Under this revised model, increasing generation length incurs a substantially higher cost than scaling model size; consequently, the tradeoff between model capacity and token budget shifts meaningfully. For instance, under the linear *LP* model, the cost of generating 8K tokens with a 14B model (which is usually insufficient to solve complex tasks) is treated as equivalent to generating 24K tokens with a 4B model (sufficient to complete most tasks). However, under the L^2D model, the same 14B@8K generation is only comparable in cost to a 4B@9K generation. This tighter bound makes it much harder for smaller models to compensate for their limited capacity through extended generation alone. Thus, only if the gap in model capacities is small enough (e.g., 32B only improves the accuracy



(a) KV v.s. Parameters (b) Iso-eFLOPs (c) Iso-FLOPs

Figure 11: Explanation of the New Scaling Law. Left: Analysis across four LLM families reveals a consistent trend of disproportionately slower KV memory growth relative to model size. For the Qwen3 series in particular, doubling model parameters results in only a $1.18\times$ increase in KV cache size. Middle and Right: We compare the Iso-Cost landscapes under the proposed cost model (b) and the traditional model (c).

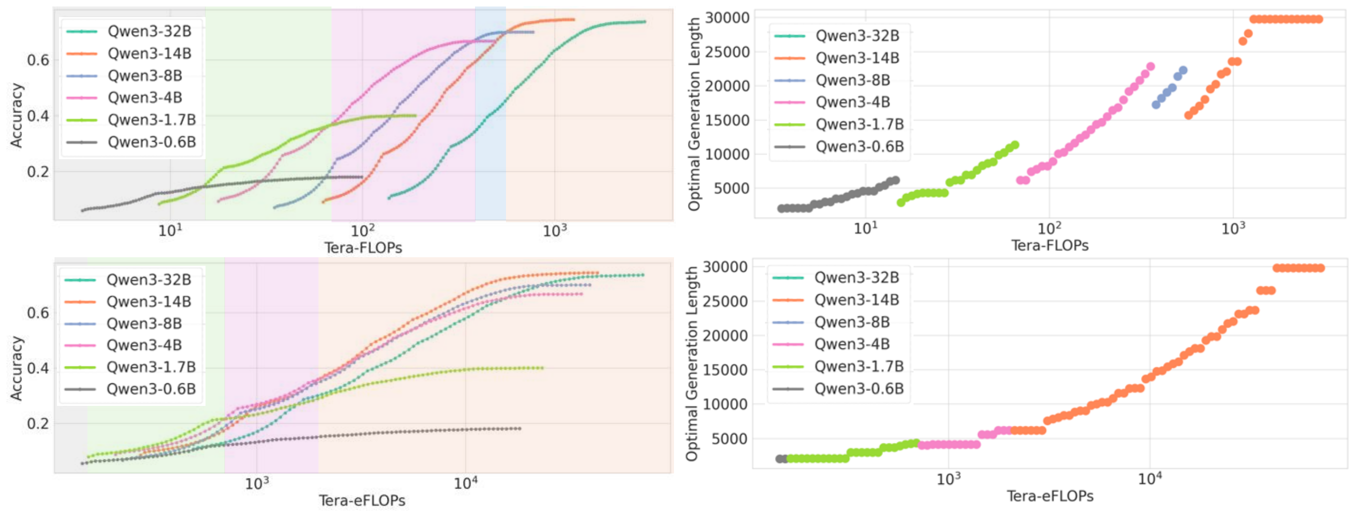


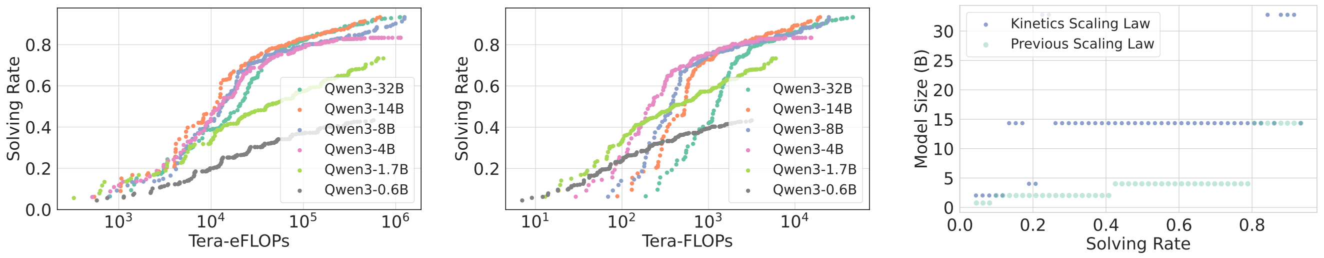
Figure 12: AIME25 Pareto Frontier (Long-CoTs). We conduct the same experiments as Figure 3.

by 3% on AIME24 compared to 14B), the benefits of extending generation length might be more effective than directly enlarging model parameters.

Figures 11b and 11c show an Iso-Cost analysis comparing two cost models. Under Kinetics Scaling Law, the cost grows quadratically with L_{out}, while the KV cache scales sublinearly with model parameters P. As a result, when total budget is low, the Iso-eFLOPs contours tend to stretch horizontally, favoring larger model sizes over longer generation lengths. This implies that increasing model size is a more efficient use of resources than generating longer outputs. In contrast, the traditional FLOPs-based model leads to steeply vertical contours, encouraging longer generation before increasing model size.

B.3 Additional Benchmarks

We evaluate on AIME25 in Figures 12 and 13a to 13c and LiveCodeBench[3]in Figures 14 and 15a to 15c (excluding the 0.6B model), following the setting described in Section 3. The empirical results support the Kinetics Scaling Law: across both benchmarks, the 0.6B and 1.7B models are consistently less effective, and the Pareto frontier is almost always dominated by the 14B models.



(a) Accuracy (eFLOPs) (b) Accuracy (FLOPs) (c) Optimal Models

Figure 13: AIME25 Score Curve (Best-of-N). We conduct the same experiments as Figures 10a to 10c.

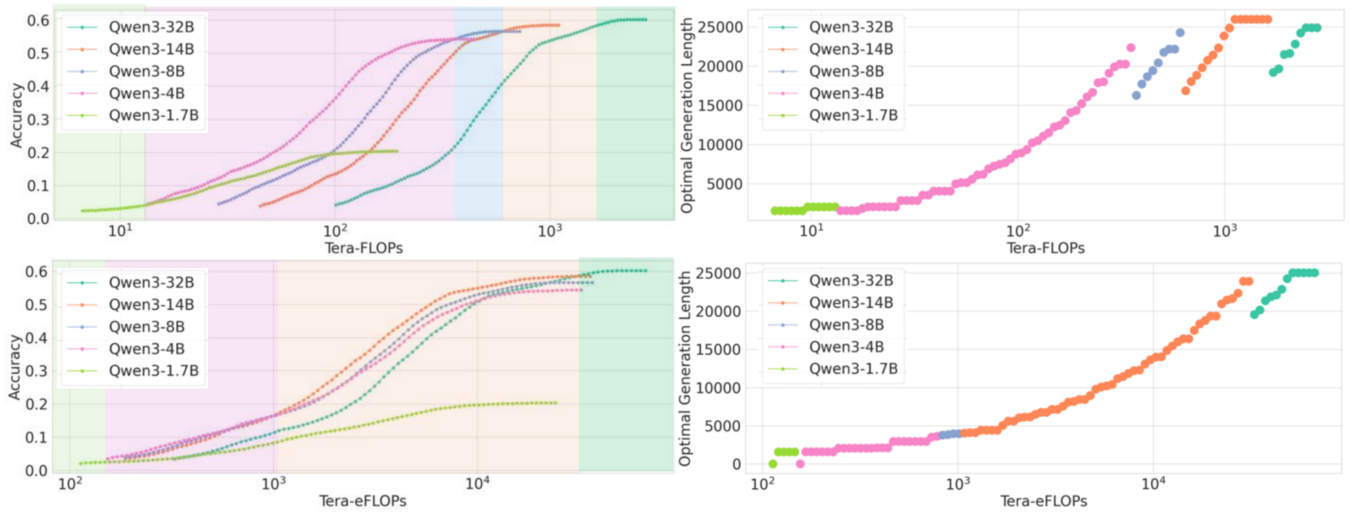
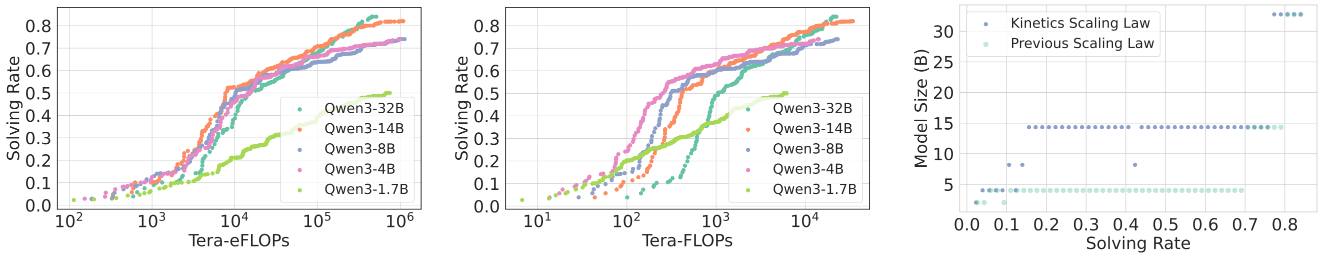


Figure 14: LiveCodeBench Pareto Frontier (Long-CoTs). We conduct the same experiments as Figure 3.



(a) Accuracy (eFLOPs) (b) Accuracy (FLOPs) (c) Optimal Models

Figure 15: LiveCodeBench Score Curve (Best-of-N). We conduct the same experiments as Figures 10a to 10c.

B.4 Additional Reasoning Models

In Figures 16 and 17a to 17c, we evaluate DeepSeek-R1 Distilled Qwen models (abbreviated as DS models) [27] on AIME24. The DeepSeek series models further demonstrate that previous scaling laws—those based on FLOPs—significantly overestimate the effectiveness of the 1.5B model. As predicted by the Kinetics Scaling Law, increasing the number of generated tokens for the 1.5B model is less effective than scaling up the model size, such as using the 7B or larger variants.

[3]For LiveCodeBench dataset, we have sampled 50 examples from the *v5* subset consisting 167 examples. Our subset comprises 24 hard, 16 medium and 10 easy examples respectively.

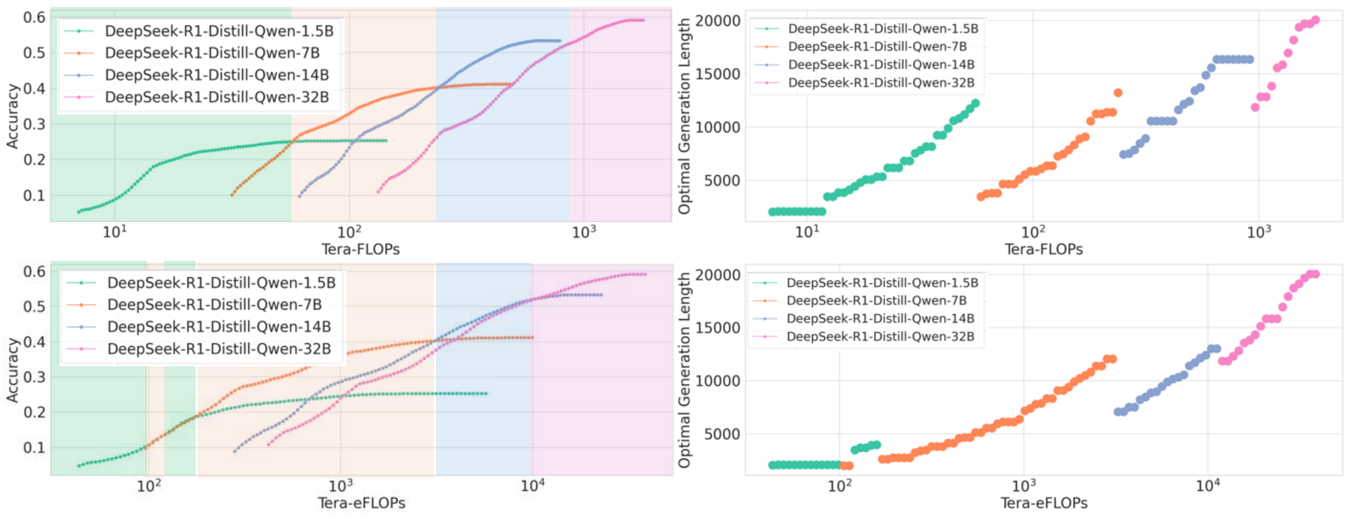
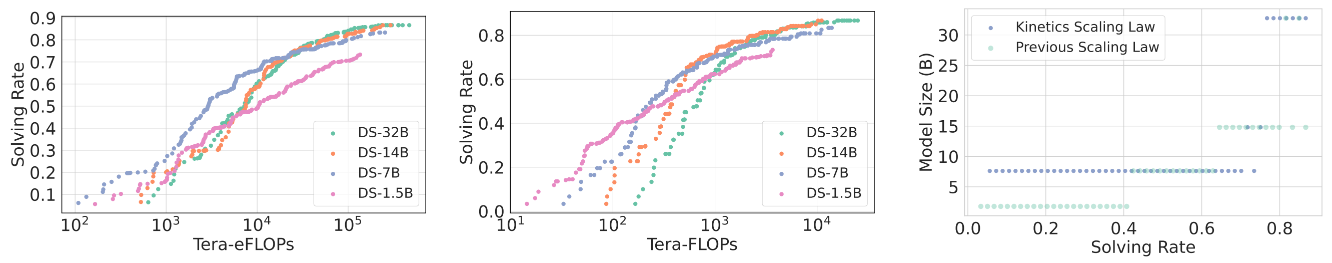


Figure 16: AIME24 Pareto Frontier (Long-CoTs). We conduct the same experiments as Figure 3 on DeepSeek Distilled Qwen series.



(a) Accuracy (eFLOPs) **(b) Accuracy (FLOPs)** **(c) Optimal Models**

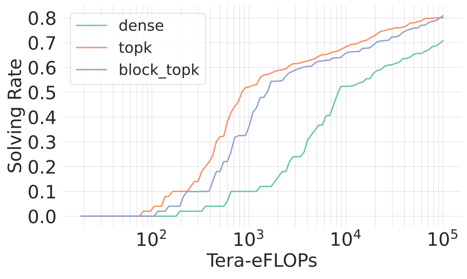
Figure 17: AIME24 Score Curve Envelope (Best-of-N). We conduct the same experiments as Figures 10a to 10c on DeepSeek Distilled Qwen series.

Interestingly, we observe a shift in the emerging model size: unlike Qwen3, where the 14B model dominates, the 7B model becomes the dominant choice in the DeepSeek series. In Figures 16, 17a and 17c, the 7B model spans most of the Pareto frontier, and Figure 16 shows that 7B models with long CoTs are more efficient and effective than 14B models with short generations. We attribute this to an architectural outlier in the DeepSeek-R1 (Qwen2.5) model series. As shown in Table 2, the DeepSeek-R1 7B model is significantly more KV memory-efficient than the Qwen3-8B model. Unlike most model series illustrated in Figure 11a, where KV cache size typically grows sublinearly with respect to model parameters, DeepSeek-R1 shows a deviation from this trend: the 14B model has approximately 3.4× more KV memory than the 7B model, while having only 2× more parameters.

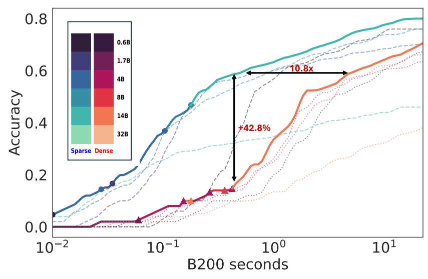
Table 2: KV memory Size for Qwen3 and DeepSeek-R1 Distilled models (per 32K tokens, unit: GB).

Qwen3	Qwen3-1.7B 3.5	Qwen3-8B 4.5	Qwen3-14B 6	Qwen3-32B 8
DeepSeek	DS-1.5B 0.875	DS-7B 1.75	DS-14B 6	DS-32B 8

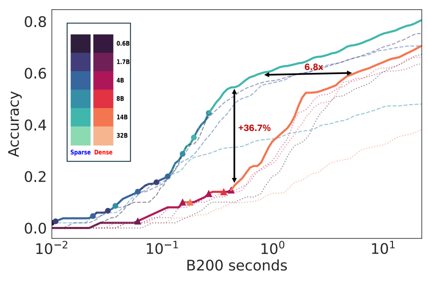
This finding highlights the importance of concrete model architecture design, rather than focusing solely on the number of model parameters. Whether KV memory size is directly related to reasoning performance remains an open question, which we leave for future investigation.



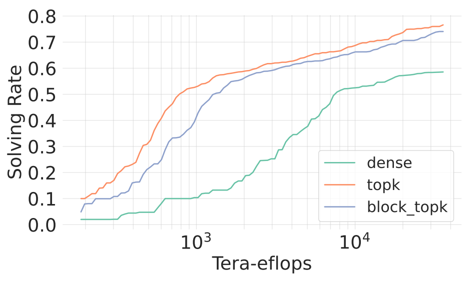
(a) Best-of-N Scaling Comparison



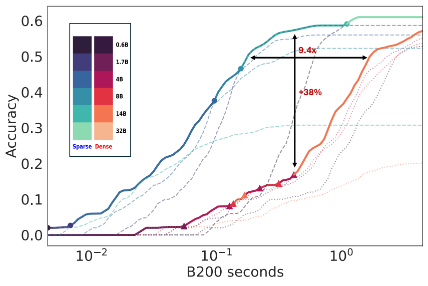
(b) *Best-of-N* Top-K Sparse Scaling



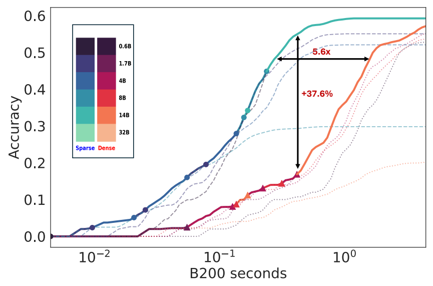
(c) Best-of-N Block Top-K Scaling



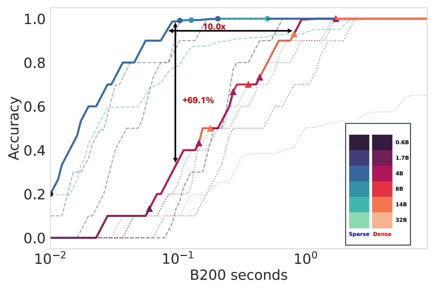
(d) Long-CoTs Scaling Comparison



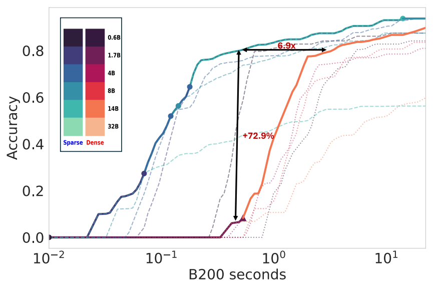
(e) Long-CoTs Top-K Sparse Scaling



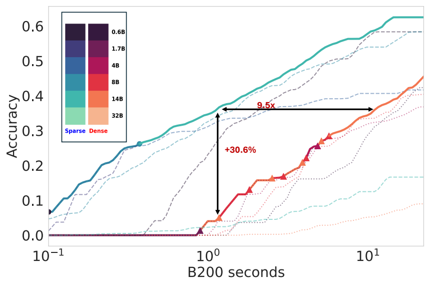
(f) Long-CoTs Block Top-K Scaling



(g) Best-of-N Scaling (Easy)



(h) Best-of-N Scaling (Medium)



(i) Best-of-N Scaling (Hard)

Figure 18: LiveCodeBench Sparse Scaling. We evaluate sparse scaling laws for Qwen3-14B model using oracle top-k and block-top-k attention on the LiveCodeBench dataset. (a)(d) compare block-top-k and oracle top-k with dense scaling under *Best-of-N* and *long-CoT* TTS settings. (b)(e) show cost-accuracy trade-offs for top-k attention. (c)(f) show trade-offs for block-top-k attention. (g)(h)(i) compare the oracle top-k scaling for easy, medium and hard difficulty questions.

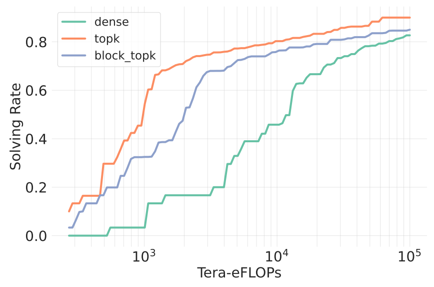
C Sparse Scaling Law

We present how we find the Pareto frontier of sparse attention models through an optimal resource allocation, which demonstrates the upper bound of scalability of a certain sparse attention algorithms. Then we present additional results supporting the kinetics sparse scaling law across multiple tasks and demonstrate how these insights enable scalable test-time scaling with sparse attention.

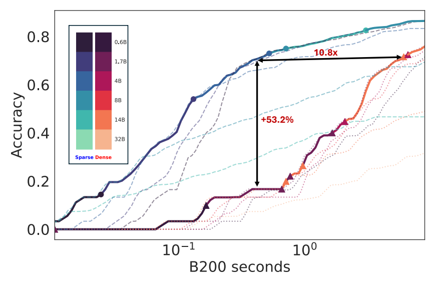
C.1 Optimal Resource Allocation with Sparse Attention Models

Problem statement. Let $\mathcal{A}$ denote the corresponding sparsity patterns (e.g., top-k, block sparse and local. Our goal is to explore the optimal tradeoff among three factors: model M, KV budget B, and number of trials, and the maximum generation length (N, n). Specifically,

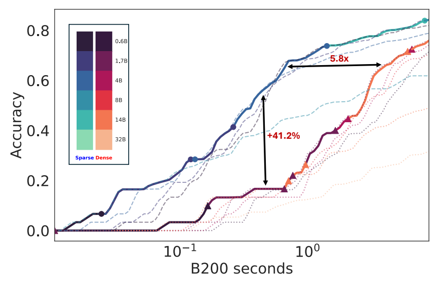
$$(N,n)_*, M_*, B_* = \arg\max_{(N,n),M,B} \ \mathrm{Acc}(N,n,B,\mathcal{A},M;T) \quad \text{s.t.} \quad C_{\mathrm{TTS}}(N,n,B,\mathcal{A},M;T) \leq C \tag{9}$$



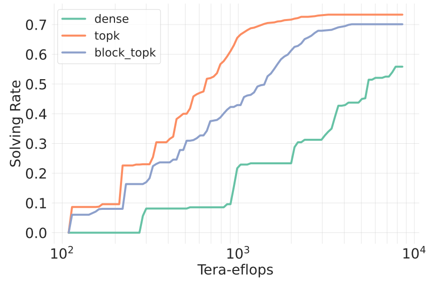
(a) *Best-of-N* Scaling Comparison



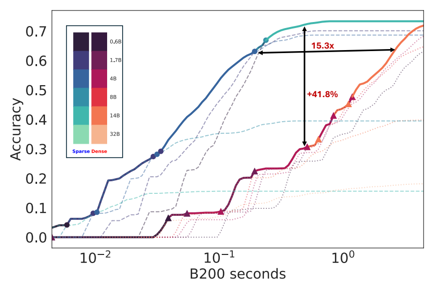
(b) *Best-of-N* Top-K Sparse Scaling



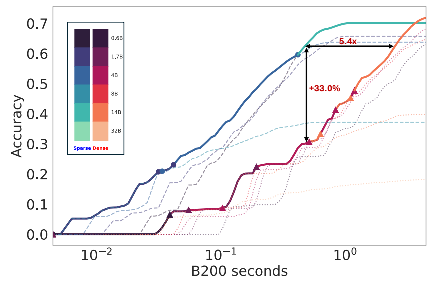
(c) *Best-of-N* Block Top-K Scaling



(d) *Long-CoTs* Scaling Comparison



(e) *Long-CoTs* Top-K Sparse Scaling



(f) *Long-CoTs* Block Top-K Scaling

Figure 19: AIME25 Sparse Scaling. We evaluate sparse scaling laws for Qwen3-14B model using oracle top-k and block-top-k attention on the AIME25 dataset. (a)(d) compare block-top-k and oracle top-k with dense scaling under *Best-of-N* and *long-CoT* settings. (b)(e) show cost-accuracy trade-offs for oracle top-k attention. (c)(f) show trade-offs for block-top-k attention.

C.2 Greedy Algorithm for Optimal Resource Allocation

We present a method to optimally schedule generation parameters (N, n) and the KV budget B for each task, establishing an upper bound on achievable performance and enabling analysis of the core tradeoff between TTS strategies and sparsity. We begin by solving the subproblem for each individual task T[4]:

$$\max \quad \text{Acc}(N_T, n_T, B_T, \mathcal{A}, M; T) \quad \text{s.t.} \quad C_{\text{TTS}}(N_T, n_T, B_T, \mathcal{A}, M; T) \leq C \tag{10}$$

Empirically, we discretize the searching space. For instance, in *Best-of-N*, we discretize the space of N and B by producing a search grid:

$$G = \{N_0, N_1, \ldots, N_i\} \otimes \{B_0, B_1, \ldots, B_j\}$$

For each pair $(N_a, B_b) \in G$, we compute the corresponding cost $C_{T,(a,b)}$ and accuracy $\text{Acc}_{T,(a,b)}$. We use $(N_T, B_T) \in G$ which maximizes the accuracy under the cost constraint C as an approximation for Equation (10). By combining the optimal configurations (N_T, B_T) for all tasks T, we obtain a solution to the overall problem in Equation (9). Similar discretizations also applies for *Long-CoTs*. Thus we find the optimal resource allocation.

We describe the procedure for identifying optimal resource allocations and establishing the Pareto frontier for sparse attention models in Algorithms 1 and 2, as a supplement to Appendix C.1. Given a fixed cost constraint C, we perform a grid search over key parameters: KV budgets and either reasoning trials or maximum generation lengths.

Empirically, we sweep over KV budgets {32, 64, 128, 256, 512, 1024}; reasoning trials {1, 2, 4, 8, 16, 32} (with a reduced upper limit for the 14B and 32B models to save computation time); and generation lengths {2k, 4k, 6k, 8k, 10k, 12k, 14k, 16k, 18k, 20k, 22k, 24k, 26k, 28k, 30k, 32k}.

It is important to note that we do not consider inter-request resource scheduling strategies, such as early stopping or dynamic reallocation across requests [23], since we aim to ensure fairness across all inputs. Instead, the cost constraint C is interpreted as the maximum allowable cost per request (not the average), even if some requests achieve saturated accuracy below that threshold.

C.3 Additional Benchmarks

Beyond AIME24, we evaluate our approach on LiveCodeBench [34] and AIME25 [51]. LiveCodeBench features complex programming problems from recent coding contests, while AIME25 consists of challenging math problems. In both cases, sparse attention—particularly

[4]For fairness, we do not schedule resources across tasks, but consider a resource upper bound for all the tasks.

Algorithm 1: Best-of-N optimal resource allocation under cost C

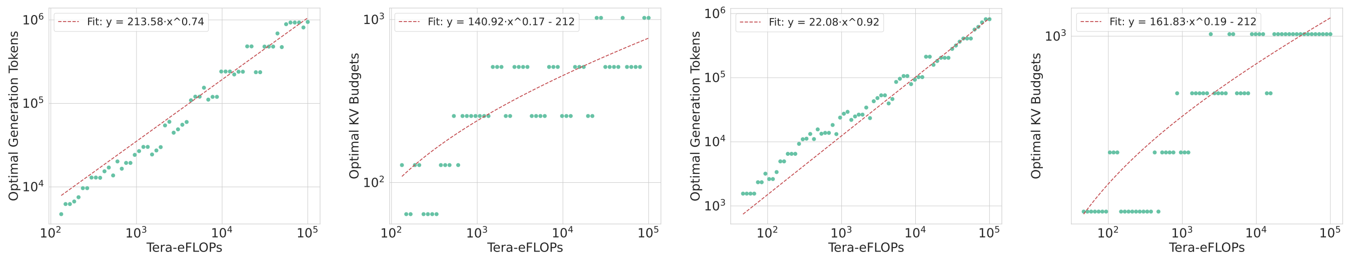
```
Data: Tasks $\mathcal{T}$, KV budgets $\{B_1, \dots, B_j\}$, trial counts $\{N_1, \dots, N_i\}$, cost limit $C$
Result: Average of maximum accuracy per task under cost $C$
AccumBestAcc ⟵ 0  Count ⟵ 0;
for task $T$ in $\mathcal{T}$ do
    for KV budget $B_b$ do
        Generate $S \geq \max\{N_1, .., N_i\}$ responses using $B_b$ for task $T$;
        for trial count $N_a$ do
            compute cost $c_{b,a}^{(T)}$;
            if $c_{b,a}^{(T)} \leq C$ then
                Compute accuracy $\mathrm{Acc}_{b,a}^{(T)}$ = Pass@$N_a$;
                if $Acc_{b,a}^{(T)} > BestAcc$ then
                    BestAcc ⟵ $\mathrm{Acc}_{b,a}^{(T)}$;
                end if
            end if
        end for
    end for
    AccumBestAcc += BestAcc; Count += 1;
end for
AvgBestAcc = AccumBestAcc/Count;
return AvgBestAcc;
```

Algorithm 2: Long-CoTs optimal resource allocation under cost C

```
Data: Tasks $\mathcal{T}$, KV budgets $\{B_1, \dots, B_j\}$, gen. lengths $\{n_1, \dots, n_i\}$, samples $S$, cost limit $C$
Result: Average of maximum accuracy per task under cost $C$
AccumBestAcc ⟵ 0  Count ⟵ 0;
for task $T$ in $\mathcal{T}$ do
    BestAcc ⟵ 0;
    for gen. length $n_a$ do
        for KV budget $B_b$ do
            Generate $S$ responses using $(B_b, n_a)$; compute cost $c_{b,a}^{(T)}$;
            if $c_{b,a}^{(T)} \leq C$ then
                Compute accuracy $\mathrm{Acc}_{b,a}^{(T)}$ = Pass@1;
                if $Acc_{b,a}^{(T)} > BestAcc$ then
                    BestAcc ⟵ $\mathrm{Acc}_{b,a}^{(T)}$;
                end if
            end if
        end for
    end for
    AccumBestAcc += BestAcc; Count += 1;
end for
AvgBestAcc = AccumBestAcc/Count;
return AvgBestAcc;
```

oracle top-k—consistently outperforms dense attention. Block top-k attention, a tractable alternative, closely matches the performance of the oracle.

For LiveCodeBench, we sample 50 problems from the *v5* subset (24 hard, 16 medium, 10 easy). As shown in Figure 18, oracle top-k attention can achieve $\sim 10\times$ speedup in high-accuracy regimes and improves coverage by 40–50% in low-cost regimes. Conversely, the



(a) AIME25 Gen. **(b) AIME25 Budget** **(c) LiveCodeBench Gen.** **(d) LiveCodeBench Budget**

Figure 20: Tradeoff Between Generated Tokens and KV Budget. We empirically characterize the tradeoff between increasing generation length and allocating a larger KV cache budget using Qwen3-8B. For AIME25 ((a)(b)) and LiveCodeBench ((c)(d)), we identify the optimal KV budget and generated tokens (defined as number of reasoning trials times the average generated tokens per trial) to achieve the highest problem-solving rate under every cost constraint C.

tractable alternative, Block top-k yields 5–6× speedup and 30–40% coverage gains. We further show how the benefits of sparse attention scale with problem difficulty (Figures 18g to 18i).

Figure 19 confirms similar trends for AIME25, with substantial gains in both accuracy and efficiency under sparse attention.

C.4 Additional Analysis

Fixing a model (e.g., Qwen3-8B), we investigate the tradeoff between generating more tokens through Best-of-N and increasing the KV budget in Figures 20a to 20d. As the figures suggest, on AIME25, each doubling of total compute cost increases the optimal KV budget by 1.13×, while generated tokens grow by 1.67×; on LiveCodeBench, these factors are 1.14× and 1.89×, respectively. We find that although the concrete numbers depend on the types of tasks, the overall results confirm our suggestions in the main paper that allocating compute toward generating more responses is generally more effective than expanding KV budget, highlighting the scalability of sparse attention.

D Experimental Details

In this section, we explain the details about our experiments.

D.1 Estimate Cost, Accuracy and Solving Rate

When empirically measuring cost, one major challenge is the difficulty of controlling the actual generation length. Although it is possible to set an upper bound on the number of generated tokens, there is no guarantee that the model will utilize the full budget. For instance, in our Best-of-N experiments, we set the maximum number of generated tokens to 32,768, yet the average generation length was only 14K–16K tokens.

Furthermore, it is important to model the relationship between actual inference cost and performance metrics, such as accuracy in Long-CoTs or solving rate in Best-of-N. Relying solely on the maximum allowed generation length to estimate cost can substantially underestimate the efficiency of models that solve problems with much shorter responses—an ability that **may** reflect higher capability.

To address this challenge, we first sample S independent reasoning traces $r_1, r_2, \ldots, r_S$ from model M on task T, with the maximum allowed number of tokens set to n. We slightly generalize Equation (6) as:

$$\begin{aligned} C_{\text{TTS}} &= 2NP\mathbb{E}[L_{\text{out}}] + 2rNL_{\text{in}}D\mathbb{E}[L_{\text{out}}] + rND\mathbb{E}[L_{\text{out}}^2] \\ &\quad + 2IL_{\text{in}}D\mathbb{E}[L_{\text{out}}] + IND\mathbb{E}[L_{\text{out}}^2] \\ &= a\mathbb{E}[L_{\text{out}}] + b\mathbb{E}[L_{\text{out}}^2] + c, \end{aligned} \tag{11}$$

where a, b, and c are constants determined by the model architecture and test-time strategies (e.g., the value of n). The expectations are estimated from the sampled traces, whose distribution is influenced by the model M, the token limit n, and the task T.

For Long-CoTs, we fix $N = 1$ in Equation (11) and vary n. From the sampled traces, we estimate the accuracy (Pass@1), and compute the corresponding cost by substituting the empirical values of $\mathbb{E}[L_{\text{out}}]$ and $\mathbb{E}[L_{\text{out}}^2]$ measured under each n.

For Best-of-N, we fix $n = 32{,}768$, and estimate the solving rate (Pass@K) following the methodology of Brown et al. [3]. The corresponding cost is then computed by substituting $N = K$ into Equation (11).

Similarly, we can estimate the cost for sparse attention models using Equations (7) and (8).

Advanced control of generation lengths is an active area of research [48, 56, 78], but it is beyond the scope of this paper.

D.2 Top-K Attention and Block Top-K Attention

In this section, we explain the sparse attention algorithms discussed in the main paper, namely *Top-K Attention* and *Block Top-K Attention*.

During the decoding phase of a large language model (LLM), the self-attention mechanism computes a weighted average of past values as follows:

$$o = \text{Softmax}\left(\frac{qK^\top}{\sqrt{d}}\right)V = wV, \quad q \in \mathbb{R}^{1\times d}, \quad K, V \in \mathbb{R}^{n\times d}, \quad w \in \mathbb{R}^{1\times n}, \tag{12}$$

where d is the head dimension and n is the context length. The key and value matrices are given by $K = [k_1, k_2, \ldots, k_n]$, $V = [v_1, v_2, \ldots, v_n]$, where each $k_i, v_i \in \mathbb{R}^{1\times d}$ are cached from previous decoding steps.

Top-K Attention. Top-K Attention is a sparsification method where only the K most relevant tokens (i.e., those with the highest attention scores) are selected to compute the output. Formally, instead of computing the full softmax, we define a sparse attention weight vector:

$$w_i = \begin{cases} \frac{\exp(s_i)}{\sum_{j\in\mathcal{I}_K}\exp(s_j)} & \text{if } i \in \mathcal{I}_K, \\ 0 & \text{otherwise,} \end{cases} \quad \text{where} \quad s_i = \frac{qk_i^\top}{\sqrt{d}}, \quad \mathcal{I}_K = \text{TopK}_K(s), \tag{13}$$

Here, $\mathcal{I}_K$ denotes the indices of the top K attention scores s_i. By masking out the less important positions, this approach reduces the computational and memory cost of attention from $O(n)$ to $O(K)$, where $K \ll n$.

Block Top-K. Block Top-K Attention is a block-level sparse attention mechanism. Instead of selecting individual tokens based on attention scores, this method selects entire blocks of tokens, thereby reducing the number of attention computations.

Specifically, assume the full sequence of n keys is divided into $m = \frac{n}{\texttt{BLOCK\_SIZE}}$ consecutive blocks, each of size `BLOCK_SIZE`:

$$K = [k_1, \ldots, k_n] \to \{K_1, K_2, \ldots, K_m\}, \quad K_i \in \mathbb{R}^{\texttt{BLOCK\_SIZE}\times d}$$

For each block K_i, we first compute the average key vector:

$$\bar{k}_i = \frac{1}{\texttt{BLOCK\_SIZE}} \sum_{j=1}^{\texttt{BLOCK\_SIZE}} k_{i,j}$$

Next, we compute the attention score between the query q and each block's average key:

$$s_i = \frac{q\bar{k}_i^\top}{\sqrt{d}}, \quad \text{for } i = 1, 2, \ldots, m$$

We then select the top $K' = \frac{K}{\texttt{BLOCK\_SIZE}}$ blocks based on the scores s_i, denoted by the index set $\mathcal{J}_{K'} = \text{TopK}_{K'}(s)$. Attention is computed only over the tokens within the selected blocks. The sparse attention weights are defined as:

$$w_i = \begin{cases} \frac{\exp(s_i)}{\sum_{j\in\mathcal{I}_K}\exp(s_j)} & \text{if } i \in \mathcal{I}_K \subseteq \text{tokens in selected blocks,} \\ 0 & \text{otherwise} \end{cases}$$

For both algorithms, K is the KV budget. For GQA, we conduct an average pooling across all the query heads in a group, ensuring that the total number of retrieved key-value vectors does not exceed the allocated KV budget.

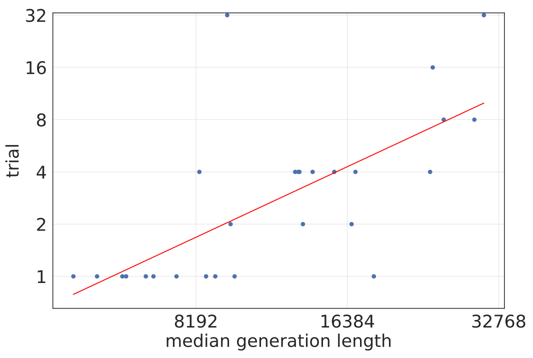
Implementation. Here we provide details of our block top-k attention implementation. We build our inference backend on Flashinfer [82], incorporating support for paged attention [39] and continuous batching [83]. Alongside the paged KV cache, we introduce an auxiliary data structure to store block-level average key vectors. The KV block size is chosen such that the memory load from the block-average vectors and the selected top-k KV blocks remains balanced. This design enables sub-quadratic KV loading cost as the number of reasoning tokens increases.

E Related Work

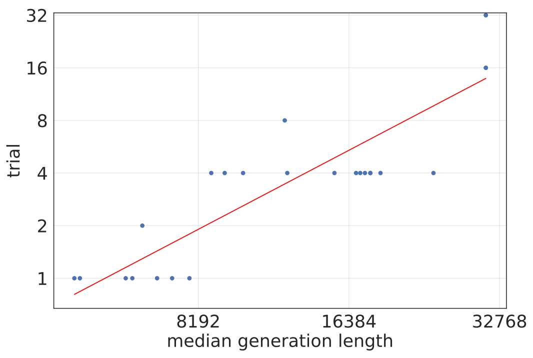
Efficient Attention. Sparse attention [4, 5, 9, 10, 38, 41, 58, 77, 84, 86, 88] has been comprehensively studied to reduce the attention cost when processing long sequeces. In parallel, approaches like FlashAttention [13, 15] accelerate attention by maximizing hardware efficiency. To address the quadratic complexity of standard attention, researchers have also explored linear attention architectures [11, 25, 26, 37]. Additionally, quantization and low-precision methods [30, 44, 47] have been broadly applied for improving inference efficiency.

Efficient Inference. Orca [83], vLLM [39], and SGLang [89] are widely adopted to enhance the efficiency of LLM serving. Our analysis builds on the practical designs and implementations of these systems. In parallel, speculative decoding [6, 40, 52, 61] has been proposed to mitigate the memory-bandwidth bottleneck during LLM decoding. Additionally, model compression and offloading [17, 22, 43, 63, 68] techniques are playing a crucial role in democratizing LLM deployment.

Efficient Test-time Strategies. Optimizing reasoning models to generate fewer tokens has been shown to directly reduce inference-time cost [2, 49, 71]. Recent work such as CoCoNut [28] and CoCoMix [69] explores conducting reasoning in a latent space, thereby reducing decoding time. Methods like ParScale [8], Tree-of-Thoughts [80], and Skeleton-of-Thoughts [59] aim to improve efficiency by enabling parallel reasoning. Architectural innovations such as CoTFormer [54] further enhance efficiency by adaptively allocating computational resources across tokens. Efficient reward-model-based [64, 67, 76] test-time scaling algorithms are also comprehensively studied.



(a) gen length vs N_{opt} correlation (top-k attention)



(b) gen length vs N_{opt} correlation (block top-k attention)

Figure 21: Correlation between Generation Length and Number of Trials. Longer generations correlate strongly with the optimal number of trials (N_{opt}), serving as a proxy for problem difficulty. (a) shows this trend for top-k and block top-k attention on the AIME24 dataset using the Qwen3-8B model.

F Limitations, Future Scope, and Broader Impact

Limitations. Our experiments primarily focus on **Qwen3** [78] and **DeepSeek-R1-Distilled-Qwen** [27], two state-of-the-art pretrained reasoning model series, evaluated from the inference perspective. However, the effects of training and post-training strategies are not fully explored and may influence the performance gaps and robustness to sparse attention mechanisms. In addition, our cost analysis assumes a cloud-based serving environment, where computational resources are typically sufficient and large batch sizes are feasible. In contrast, local deployment scenarios, such as those using Ollama[5], often face limited VRAM where access to model parameters can dominate inference costs. Smaller models may be more appropriate in such settings, and our findings may not fully extend to these use cases.

Future Scope. Our sparse scaling law offers valuable insights for enriching the applications of sparse attention algorithms and the design space of test-time scaling strategies. On one hand, except for top-k, currently we only discuss a simple variant, i.e., block top-k, and have already demonstrated strong scalability. More advanced sparse attention algorithms [9, 42, 70, 84] are emerging these days. We do believe they can eventually push the scalability of test-time scaling to a much higher boundary. On the other hand, test-time scaling algorithms are proposed to adaptively allocate computation to tasks, or even to tokens [2, 48, 49, 54]. Extending them towards to new resource allocation problems in sparse attention is critical to reach the limit of Kinetics sparse scaling law. For instance, since generation length strongly correlates with the optimal number of trials under sparse attention (as shown in Figure 21), it can be used as a dynamic signal to adjust the number of trials and KV budget. Moreover, sparse attention drastically reduces inference cost, enabling more reasoning trials and longer generations. This unlocks greater flexibility in configuring TTS strategies within a fixed resource budget.

Broader Impact. This work aims to contribute to the understanding of efficiency and scalability challenges in the test-time scaling era, spanning model architecture, system-level implementation, and hardware design. We highlight the central role of sparsity in addressing these challenges. Our study is algorithmic in nature and does not target specific applications. While large language models can be misused in harmful ways, this work does not introduce new capabilities or risks beyond those already present in existing systems. Test-time scaling can consume a substantial amount of energy, raising concerns about the environmental sustainability of widespread deployment. By promoting sparse attention, our work hopes to help to reduce the carbon footprint and energy consumption of inference systems and support the broader goal of sustainable AI.

[5]https://github.com/ollama/ollama