

GPU Kernel Optimization for Efficient Sparse Attention and Top-K Selection in Large Language Models

Daphne Bruggeman

Department of Electrical Engineering and Computer Science, University of Kansas, Lawrence, KS, USA.

daphnebruggeman00@ku.edu

Ann-Christin Granstrom

Department of Computer Science, University of Alabama at Birmingham, Birmingham, AL, USA.

annchristin2001@uab.edu

Abstract

Large language models have become central to contemporary artificial intelligence systems, yet their inference and training costs are increasingly dominated by attention mechanisms and auxiliary selection operations. Sparse attention and top-k selection offer promising pathways for reducing computation, memory traffic, and energy consumption, but their benefits depend heavily on how they are mapped onto GPU hardware. This paper examines GPU kernel optimization for efficient sparse attention and top-k selection from a system-level perspective. It argues that the central challenge is not merely reducing arithmetic operations but managing irregular memory access, synchronization, load imbalance, and dynamic control flow across the memory hierarchy. The paper analyzes architectural trade-offs among fixed, learned, and dynamic sparsity; row-wise selection kernels; quantization and low-precision formats; and serving infrastructure such as paged memory, batching, and scheduling. It further considers robustness, fairness, governance, and sustainability implications, because optimized kernels can shape which models are economically viable and which application domains receive low-latency service. The discussion highlights that kernel-level decisions have second-order effects on auditability, reproducibility, and equitable deployment. The paper concludes that efficient sparse attention and top-k selection should be treated as socio-technical infrastructure problems rather than isolated micro-optimizations. Sustainable progress requires co-design across algorithms, compilers, hardware, serving systems, and policy frameworks, with performance claims accompanied by transparent evaluation of accuracy, energy use, and distributional impact.

Keywords

sparse attention, top-k selection, GPU kernels, large language models, inference systems, memory hierarchy, sustainability, AI governance

1 Introduction Large language models have reshaped natural language processing and adjacent fields by scaling transformer architectures to billions of parameters and long context windows [1]. This scaling has produced remarkable capabilities, but it has also exposed a fundamental systems bottleneck: attention computation grows quadratically with sequence length, while top-k selection operations appear throughout routing, retrieval, mixture-of-experts gating, and approximate attention. As a result, modern inference systems are constrained less by peak arithmetic throughput than by memory

bandwidth, cache behavior, synchronization overhead, and the irregularity of sparse data movement [2]. The problem is therefore not simply one of algorithmic sparsity but of GPU kernel optimization and system architecture. Sparse attention attempts to reduce the effective number of token interactions by retaining only a subset of attention links, either through fixed patterns, learned routing, or dynamic query-dependent selection [5], [6]. Top-k selection is often the mechanism that identifies the most relevant links or experts, but it introduces a difficult GPU workload: each row of a matrix may require a different number of comparisons, exchanges, and writes, and the selected indices are rarely contiguous. This irregularity conflicts with the SIMT execution model of GPUs, where threads in a warp are most efficient when they follow the same control path and access coalesced memory [14]. Consequently, kernel design must confront a tension between theoretical sparsity and practical utilization. The deployment context intensifies these challenges. Large language model serving systems must handle variable sequence lengths, batch sizes, request priorities, and latency targets [4]. Paged memory management, continuous batching, and KV cache reuse have improved throughput, but they also create new constraints on how sparse attention kernels can be scheduled [20]. A kernel that performs well in isolation may degrade under real serving conditions because of cache pollution, memory fragmentation, or synchronization with other kernels. This paper therefore treats GPU kernel optimization as part of a broader socio-technical infrastructure, where performance, robustness, fairness, and sustainability are intertwined. The expansion of large-scale model training and inference has already raised concerns about energy use, transparency, and concentrated access to computation [21], [22]. Optimizing kernels can reduce cost and environmental impact, but it can also obscure approximation errors and make auditing more difficult if selection behavior is not documented. This paper proceeds by reviewing the background of attention and selection workloads, analyzing architectural trade-offs, examining GPU kernel design strategies, and discussing system integration, governance, and sustainability. The central claim is that efficient sparse attention and top-k selection require co-design across algorithms, kernels, compilers, serving infrastructure, and policy. Improvements at the kernel level are necessary but insufficient unless they are embedded in transparent evaluation practices and deployment architectures that account for robustness and equity.

2 Background and Problem Setting

The transformer architecture relies on self-attention to compute relationships among all tokens in a sequence [1]. In dense attention, every query attends to every key, producing a quadratic computational pattern that becomes prohibitive for long contexts. Early sparse transformer designs introduced fixed patterns such as strided, local, and global attention to reduce the number of interactions while preserving some long-range connectivity [7]. Longformer combined sliding-window attention with global tokens, enabling linear scaling for long documents [5]. Big Bird further mixed random, local, and global attention, providing theoretical expressiveness while reducing cost [6]. These approaches demonstrate that sparsity can be effective, but they also reveal that fixed patterns may not align with actual query relevance. Alternative efficiency strategies replace dense attention with low-rank or kernel-based approximations. Linformer projects keys and values into a lower-dimensional space to achieve linear complexity [9]. Reformer uses locality-sensitive hashing to group similar queries and keys, reducing attention cost while attempting to preserve accuracy [8]. Linear attention methods reformulate attention as a recurrent or kernel feature map computation, trading expressiveness for efficiency [10]. Performer uses random feature approximations to estimate softmax attention kernels [11]. These methods are algorithmically elegant, but their GPU implementations often suffer from additional memory traffic, numerical instability, or limited parallelism. In practice, the boundary between algorithm and

kernel is blurred: an approximation that reduces arithmetic may increase memory pressure or synchronization, thereby reducing end-to-end performance. Top-k selection is another recurring primitive. In autoregressive decoding, speculative methods and memory-efficient attention often need to identify the most probable tokens or the most relevant keys [12]. In mixture-of-experts models, routing networks select a small number of experts per token, which requires row-wise top-k operations over gating scores. In retrieval-augmented generation, top-k selection ranks documents or passages before attention. The operation is deceptively simple: for each row, find the largest values and their indices. However, on GPUs, row-wise top-k is challenging because rows may be long, distributions may be skewed, and the selected elements must be gathered and written in an order that supports downstream computation. Naive approaches that sort entire rows waste bandwidth and compute, while approximate methods may compromise model quality. The hardware context is equally important. Modern GPUs provide high floating-point throughput, large register files, shared memory, and high-bandwidth memory, but their effective performance depends on data reuse and access patterns [15], [16]. The roofline model illustrates that many neural network operations are memory-bound rather than compute-bound, meaning that reducing arithmetic alone may not improve latency if data movement dominates [14]. Quantization and low-precision formats can alleviate memory pressure by reducing the size of weights and activations [18], [19]. Yet sparse attention and top-k selection often operate on indices and masks, which are integer-heavy and irregular. These metadata pathways can become the true bottleneck, especially when kernels repeatedly read and write masks, indices, and partial results. The problem setting is therefore multidimensional. A sparse attention algorithm may reduce floating-point operations, but it may also introduce gather operations, variable-length loops, and divergent branches. A top-k kernel may reduce the number of candidates, but it may require multiple passes over memory or costly global synchronization. Serving systems add further constraints, because kernels must be launched efficiently, overlap with data transfers, and coexist with other operations [13]. The relevant question is not whether sparsity is theoretically cheaper, but whether the complete execution path achieves lower latency, higher throughput, and acceptable accuracy under realistic deployment conditions.

3 Architectural Trade-offs in Sparse Attention and Top-K Selection

The first major trade-off concerns the granularity of sparsity. Fixed sparse patterns are predictable and can be optimized with regular memory access, but they may miss important long-range dependencies [5], [6]. Dynamic sparsity selects attention links based on query content, which can improve accuracy but creates load imbalance and irregular memory access. Learned sparsity, where a model or routing network predicts which links to keep, offers a middle ground, but it requires additional parameters, training complexity, and inference overhead. The choice among these options affects not only model quality but also kernel design, because fixed patterns can be unrolled or tiled, whereas dynamic patterns require sorting, selection, and compaction. Top-k selection adds a second trade-off between exactness and speed. Exact top-k algorithms guarantee that the selected elements are the true largest values, which is important for reproducibility and auditability. Approximate top-k methods may use sampling, thresholds, or hierarchical pruning to reduce work, but they can produce unstable selections [17]. In mixture-of-experts routing, unstable top-k can cause tokens to be routed inconsistently, affecting both accuracy and fairness. In retrieval-augmented generation, approximate selection may omit relevant evidence, leading to hallucination or biased responses. The governance implication is that approximate kernels should be documented with the same seriousness as approximate model compression, because they alter system behavior [23]. A third trade-off involves memory layout and metadata representation. Sparse attention often

stores masks, indices, or block tables alongside dense tensors. If these metadata structures are not compact, the memory savings from sparsity can be negated. Top-k selection produces indices that must be gathered from key or value caches, and the gather pattern may be scattered across memory. On GPUs, scattered reads reduce effective bandwidth and increase latency. Therefore, kernel designers must consider whether to use block-sparse representations, compressed sparse row formats, or bitmap masks. Each representation favors different access patterns and different levels of parallelism. The best choice may depend on sequence length, batch size, and the degree of sparsity. A fourth trade-off concerns precision. Sparse attention scores and top-k comparisons can be performed in lower precision to reduce memory traffic, but low-precision comparisons may alter rankings when values are close [18], [19]. Quantization can introduce ties or near-ties that change selected indices, which may have cascading effects in downstream layers. Mixed-precision strategies can preserve ranking stability by using higher precision for selection and lower precision for value aggregation, but this adds complexity and may reduce the benefit of quantization. The system designer must therefore evaluate not only average accuracy but also ranking stability, tail behavior, and sensitivity across input distributions. Finally, there is a trade-off between specialization and generality. A kernel optimized for one sparsity pattern, sequence length, or batch size may outperform a general kernel, but it may not adapt well to changing workloads. Autotuning and just-in-time compilation can select among kernel variants, but they introduce compilation latency and operational complexity [13]. In production serving, a specialized kernel that is rarely used may not justify its maintenance cost. Conversely, a general kernel may leave performance on the table for high-value workloads. This tension is familiar in systems engineering: optimization improves efficiency but can increase technical debt when the system becomes difficult to understand, reproduce, or modify [24].

Governance frameworks should therefore treat kernel specialization as an architectural decision with long-term maintainability implications.

4 GPU Kernel Design for Efficient Sparse Attention and Top-K Selection

At the kernel level, efficient sparse attention begins with recognizing that attention is not a single operation. It consists of score computation, masking or selection, softmax normalization, and value aggregation. Each stage has different arithmetic intensity and memory behavior. FlashAttention demonstrated that tiling and recomputation can reduce memory traffic by keeping intermediate results in on-chip memory, thereby improving exact attention performance without approximating the attention matrix [2]. FlashAttention-2 further refined work partitioning and parallelism to improve utilization on modern GPUs [3]. These advances show that IO-awareness is often more important than arithmetic reduction. Sparse attention kernels should follow the same principle: they must minimize global memory reads and writes, not merely reduce floating-point operations. Top-k selection is a particularly demanding kernel because it is fundamentally a reduction with ordering constraints. For each row, the kernel must maintain a set of candidate values and indices, update them as new elements are scanned, and produce a sorted or unsorted list of the top k results. Row-wise top-k selection can be implemented with warp-level primitives, shared memory, or hierarchical reduction across thread blocks. The optimal strategy depends on row length, k, data distribution, and the availability of sorted inputs. Recent work on row-wise top-k selection has focused on reducing the number of passes over memory and using warp-level cooperation to handle irregular row lengths [17]. The key insight is that selection is not just a sorting problem; it is a memory-bound filtering problem that benefits from early pruning, threshold estimation, and compact data structures. Memory coalescing is central to both sparse attention and top-k selection. When threads in a warp access consecutive addresses, the GPU can service the request efficiently. When they access scattered addresses, the request is split into

multiple transactions, reducing effective bandwidth [15]. Sparse attention often gathers keys and values based on selected indices, which can produce scattered access. To mitigate this, kernels can group selected indices by block, use shared memory to stage gathered data, or reorganize the key-value cache into fixed-size pages that improve locality. Paged memory management, originally developed for serving systems, can also inform sparse attention kernels because it provides a structured way to allocate and retrieve non-contiguous memory [4]. The challenge is to align algorithmic sparsity with hardware-friendly memory layouts. Divergence and load imbalance are additional concerns. In a warp, if different threads process rows of different lengths or different numbers of selected elements, some threads may idle while others continue. This reduces warp efficiency. Techniques such as row bucketing, dynamic work assignment, and persistent kernels can help balance load across streaming multiprocessors. However, these techniques require careful synchronization to avoid races and deadlocks. They also increase register pressure and shared memory usage, which may reduce occupancy. Kernel designers must therefore tune the trade-off between parallelism and per-thread work. The roofline model remains a useful guide: if the kernel is memory-bound, increasing occupancy may help hide latency; if it is latency-bound due to synchronization, reducing synchronization may be more important [14]. Quantization and low-precision computation interact with these design choices. Quantized key-value caches reduce memory footprint and bandwidth demand, which can make sparse attention more affordable [18], [19]. However, top-k selection over quantized values may require dequantization or specialized comparison logic. If dequantization is performed repeatedly, the savings may disappear. A better approach is to design selection kernels that operate directly on quantized representations or use a two-stage process: first select candidates using low-precision approximations, then refine using higher precision. This staged approach resembles hierarchical top-k, where coarse filtering reduces the candidate set before exact selection. It can improve performance while preserving ranking accuracy, but it requires careful error analysis. Finally, kernel optimization must consider the end-to-end execution graph. Sparse attention and top-k selection are rarely isolated; they are embedded in larger models with normalization, projection, activation, and communication operations [13]. Kernel fusion can reduce memory traffic by combining operations, but it can also reduce flexibility and increase compilation complexity. Graph-level optimization can schedule kernels to overlap computation with memory transfers, but it depends on accurate performance models. The most effective designs treat sparse attention and top-k selection as part of a pipeline, not as standalone primitives. This systems perspective is essential for translating microbenchmark gains into real-world improvements.

5 System Integration, Serving, and Deployment

Infrastructures

Deploying optimized kernels in large language model serving systems introduces constraints that are absent from microbenchmarks. Production workloads have variable request rates, sequence lengths, and latency service-level objectives. Continuous batching and paged attention allow serving systems to process many requests concurrently while managing key-value cache memory [4]. These mechanisms improve throughput but create fragmented memory access patterns and dynamic batch compositions. Sparse attention kernels must therefore be robust to changing batch sizes and memory layouts. A kernel that assumes contiguous key-value storage may perform poorly when the serving system uses paged memory. Scheduling is another critical layer. Inference engines must decide which kernels to launch, in what order, and with what degree of parallelism [13]. Top-k selection may need to run before attention to identify relevant tokens, or after attention to select outputs. If top-k selection is on the critical path, its latency directly affects time to first token and inter-token latency. If it is off the critical path, it can be overlapped with other work. The placement

of selection affects the end-to-end design. For example, speculative decoding uses top-k or top-p sampling to propose candidate tokens, and the efficiency of that selection can determine whether speculation yields a net speedup [12]. Similarly, retrieval-augmented generation may perform top-k document selection before attention, making the selection kernel a gateway to the rest of the computation. Heterogeneous infrastructure adds further complexity. Models may be served across multiple GPUs, CPUs, or specialized accelerators. Communication costs can dominate when sparse attention requires gathering data from remote memory. In such settings, sparsity can reduce computation but increase communication if selected elements are distributed across devices. System designers must therefore consider locality-aware sparsity, where selection is constrained to data that is already available on the local device or in a nearby cache. This resembles edge computing trade-offs: moving less data is often more important than computing less. FlexGen demonstrated that aggressive memory management and offloading can enable large model inference on limited hardware, but it also showed that the cost of data movement can become the primary bottleneck [20]. Operational governance is also relevant at the deployment layer. Model updates, kernel updates, and hardware changes can alter system behavior in subtle ways. A new top-k kernel may be faster but select slightly different indices, leading to different outputs. Without versioning and reproducibility practices, it may be difficult to diagnose regressions or audit decisions. Technical debt accumulates when systems contain many specialized kernels whose interactions are poorly documented [24]. Organizations should therefore maintain performance and accuracy dashboards that track kernel-level changes alongside model-level metrics. This is not only an engineering concern but also a governance concern, because opaque optimization can undermine accountability.

6 Robustness, Fairness, and Governance Implications

Sparse attention and top-k selection are often justified as efficiency measures, but they also introduce approximation and selection effects that can have distributional consequences. If a model consistently ignores certain tokens, languages, or domains because they are less likely to be selected, then the efficiency gain may come at the cost of representational equity. This risk is particularly salient in multilingual or low-resource settings, where token frequencies and attention patterns may differ from the dominant training distribution. Fairness audits should therefore examine not only final outputs but also intermediate selection behavior. Which tokens are retained? Which experts are activated? Which documents are retrieved? These questions connect kernel design to broader concerns about bias and inclusion [22]. Transparency is complicated by the complexity of optimized kernels. Model cards and system documentation typically describe model architecture, training data, and intended use [23]. They rarely describe kernel-level approximations such as approximate top-k, block-sparse patterns, or quantized selection thresholds. Yet these choices can affect behavior in ways that matter for high-stakes applications. A governance framework should require documentation of efficiency-related approximations, including their expected error bounds, failure modes, and evaluation conditions. This does not mean that every kernel must be fully interpretable; rather, it means that the system should be auditable at the level of relevant behavior. If a model denies a loan or recommends medical information, the fact that a top-k kernel pruned candidates is a relevant system characteristic. Robustness is another dimension. Sparse attention may be sensitive to adversarial inputs that manipulate selection scores. If an attacker can influence which tokens are selected, they may steer attention away from safety-critical context. Top-k selection can also be gamed if an adversary inserts tokens that dominate the ranking. Defenses include robust selection criteria, diversity constraints, and fallback to dense attention for high-risk queries. However, such defenses increase cost and complexity. The trade-off between efficiency and robustness must be managed explicitly

rather than left to default kernel settings. In safety-critical deployments, it may be appropriate to use conservative sparsity and exact selection, while lower-risk applications can tolerate more aggressive approximations. Policy implications follow from these technical realities. Governments and standards bodies increasingly focus on AI transparency, accountability, and environmental impact [21], [22]. If efficiency optimizations are not documented, regulators may struggle to assess whether a system meets safety or fairness requirements. Conversely, requiring excessive documentation for every kernel optimization could slow innovation and burden smaller developers. A balanced approach might involve tiered reporting: high-level disclosure of efficiency techniques for all systems, with deeper technical audits for high-risk applications. Such an approach recognizes that kernel optimization is not merely an internal implementation detail but part of the evidence base for trustworthy AI.

7 Sustainability, Policy, and Future Directions

The sustainability case for GPU kernel optimization is strong. Large language models consume significant energy during training and inference, and the carbon footprint depends on data center location, hardware efficiency, and utilization [25]. Sparse attention and top-k selection can reduce energy use by lowering memory traffic and computation, but the net effect depends on whether they shorten runtime or simply shift bottlenecks. If a sparse kernel improves cache locality and reduces data movement, it can deliver real energy savings. If it introduces repeated gathering, synchronization, and metadata overhead, it may consume more energy than the dense operation it replaces. Therefore, energy accounting should accompany performance benchmarking. Reporting only latency or throughput can obscure the environmental consequences of optimization choices [21]. Future directions should pursue co-design across layers. Algorithm researchers can develop sparsity patterns that are both expressive and hardware-friendly. Compiler researchers can build transformations that recognize top-k selection and generate efficient row-wise kernels. Hardware architects can add primitives for selection, compaction, and ranked reductions. Serving system designers can expose scheduling hints and memory layouts that improve locality. Policy makers can encourage standardized measurement of energy, accuracy, and fairness. This co-design agenda is ambitious, but it reflects the reality that no single layer can solve the efficiency problem alone. The most impactful advances are likely to come from combinations of algorithmic sparsity, IO-aware kernels, and deployment-aware scheduling [2], [3], [20].

A second direction is adaptive sparsity. Rather than applying the same sparsity pattern to every layer or request, systems could adjust selection budgets based on input difficulty, latency targets, and risk level. Easy requests might use aggressive sparsity, while ambiguous or high-stakes requests could fall back to denser attention. Adaptive systems require fast estimators of input difficulty and robust mechanisms for switching between execution paths. They also require clear policies about when approximation is acceptable. This is a socio-technical challenge as much as an engineering one: users and regulators need to understand when a system is operating in an approximate mode and what safeguards apply.

A third direction is standardized benchmarking. Current evaluations often report kernel speedups under idealized conditions, making it difficult to compare results across papers. A shared benchmark suite should include realistic sequence lengths, batch sizes, sparsity levels, quantization settings, and hardware platforms. It should measure latency, throughput, energy, memory traffic, accuracy, and selection stability. Such a suite would help researchers identify which optimizations generalize and which are tailored to narrow conditions. It would also support policy discussions by providing comparable evidence about efficiency and sustainability. As sparse attention and top-k selection become standard components of large language model infrastructure, the field needs evaluation practices that match the complexity of the systems being built.

8 Conclusion

GPU kernel optimization for sparse attention and

top-k selection is a critical enabler of efficient large language model inference, but it cannot be reduced to a search for faster loops. The performance of these operations depends on memory hierarchy behavior, irregular access patterns, synchronization, load balance, quantization, and integration with serving infrastructure. Fixed and dynamic sparsity offer different trade-offs between predictability and accuracy. Exact and approximate top-k selection offer different trade-offs between speed and auditability. Quantization can reduce bandwidth pressure but may destabilize rankings. Serving systems can improve utilization but complicate memory locality. Each choice has downstream implications for robustness, fairness, and sustainability. The paper has argued for a system-level and socio-technical perspective. Kernel designers should work closely with algorithm developers, compiler engineers, and deployment teams to ensure that microbenchmark gains translate into end-to-end improvements. Governance frameworks should require documentation of efficiency-related approximations and their effects on selection behavior. Sustainability assessments should measure energy and carbon alongside latency and throughput. Fairness audits should examine intermediate selection patterns, not only final outputs. These practices are necessary because optimized kernels shape which models are feasible, which applications are affordable, and which users receive responsive service. Future progress will depend on co-design across algorithms, kernels, hardware, serving systems, and policy. Adaptive sparsity, standardized benchmarking, and transparent reporting are promising directions. The central conclusion is that efficient sparse attention and top-k selection are not merely implementation details; they are infrastructural decisions with technical, economic, and social consequences. A research agenda that treats them as such will be better positioned to deliver both performance and public value.

References

1. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 5998–6008.
2. Dao, T., Fu, D. Y., Ermon, S., Rudra, A., & Ré, C. (2022). FlashAttention: Fast and memory-efficient exact attention with IO-awareness. *Advances in Neural Information Processing Systems*, 35, 16344–16359.
3. Dao, T. (2024). FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations*.
4. Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., & Stoica, I. (2023). Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles* (pp. 611–626). Association for Computing Machinery.
5. Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.
6. Zaheer, M., Guruganesh, G., Dubey, A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., & Ahmed, A. (2020). Big Bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33, 17283–17297.
7. Child, R., Gray, S., Radford, A., & Sutskever, I. (2019). Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

8. Kitaev, N., Kaiser, Ł., & Levskaya, A. (2020). Reformer: The efficient transformer. In International Conference on Learning Representations.
9. Wang, S., Li, B. Z., Khabsa, M., Fang, H., & Ma, H. (2020). Linformer: Self-attention with linear complexity. arXiv preprint arXiv:2006.04768.
10. Katharopoulos, A., Vyas, A., Pappas, N., & Fleuret, F. (2020). Transformers are RNNs: Fast autoregressive transformers with linear attention. In Proceedings of the 37th International Conference on Machine Learning (pp. 5156–5165). PMLR.
11. Choromanski, K., Likhoshesterov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L., & Weller, A. (2021). Rethinking attention with performers. In International Conference on Learning Representations.
12. Shazeer, N. (2019). Fast transformer decoding: One write-head is all you need. arXiv preprint arXiv:1911.02150.
13. Pope, R., Douglas, S., Chowdhery, A., Devlin, J., Bradbury, J., Heek, J., Xiao, K., Agrawal, S., & Dean, J. (2023). Efficiently scaling transformer inference. Proceedings of Machine Learning and Systems, 5, 606–624.
14. Williams, S., Waterman, A., & Patterson, D. (2009). Roofline: An insightful visual performance model for multicore architectures. Communications of the ACM, 52(4), 65–76.
15. Jia, Z., Maggioni, M., Staiger, B., & Scarpazza, D. P. (2019). Dissecting the NVIDIA Volta GPU architecture via microbenchmarking. arXiv preprint arXiv:1804.06826.
16. NVIDIA. (2022). NVIDIA H100 Tensor Core GPU architecture [White paper]. NVIDIA Corporation.
17. Xie, X., Luo, Y., Peng, H., & Ding, C. (2025, May). RTop-K: ultra-fast row-wise top-K selection for neural network acceleration on GPUs. In International Conference on Learning Representations (Vol. 2025, pp. 81200-81214).
18. Frantar, E., Ashkboos, S., Hoeffler, T., & Alistarh, D. (2023). GPTQ: Accurate post-training quantization for generative pre-trained transformers. In International Conference on Learning Representations.
19. Dettmers, T., Lewis, M., Belkada, Y., & Zettlemoyer, L. (2022). LLM.int8(): 8-bit matrix multiplication for transformers at scale. Advances in Neural Information Processing Systems, 35, 30318–30332.
20. Sheng, Y., Zheng, L., Yuan, B., Li, Z., Ryabinin, M., Chen, B., Liang, P., Ré, C., Stoica, I., & Zhang, C. (2023). FlexGen: High-throughput generative inference of large language models with a single GPU. In Proceedings of the 40th International Conference on Machine Learning (pp. 31094–31116). PMLR.
21. Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and policy considerations for deep learning in NLP. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (pp. 3645–3650). Association for Computational Linguistics.
22. Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big? In Proceedings of the 2021 ACM

Conference on Fairness, Accountability, and Transparency (pp. 610–623). Association for Computing Machinery.

23. Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., Spitzer, E., Raji, I. D., & Gebru, T. (2019). Model cards for model reporting. In Proceedings of the Conference on Fairness, Accountability, and Transparency (pp. 220–229). Association for Computing Machinery.
24. Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., & Dennison, D. (2015). Hidden technical debt in machine learning systems. *Advances in Neural Information Processing Systems*, 28, 2503–2511.
25. Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L.-M., Rothchild, D., So, D., Texier, M., & Dean, J. (2021). Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*.