GEAR-X: Expanders for Next-Gen KV Cache Compression

Vivek Mirani
Department of Mechanical Engineering
Indian Institute of Technology Kharagpur
Kharagpur, West Bengal 721302, India
mirani.vivek5@gmail.com

Garima Bansal
Department of Agricultural and Food Engineering
Indian Institute of Technology Kharagpur
Kharagpur, West Bengal 721302, India
garimabansal.stem@gmail.com

Arindam Biswas
Polynom Research,
Paris, France.
arindam.biswas@polynom.io

Pabitra Mitra
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur, West Bengal 721302, India
pabitra@gmail.com

Amaljith E V
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur, West Bengal 721302, India
amaljith.cse@gmail.com

Abstract

Large Reasoning Models (LRMs) use Key-Value (KV) caching to speed up autoregressive decoding by reusing previously computed attention states for long contexts. However, KV caches grow linearly with sequence length, quickly saturating GPU memory and becoming a bottleneck for long-context reasoning. Prior work, such as GEAR (GEnerative Inference with Approximation Error Reduction), compresses KV caches by combining low-bit quantization, sparse outlier handling, and low-rank approximation. We propose *GEAR-X*, a drop-in modification that replaces unstructured magnitude-based outlier selection with *structured sparsity via expander graphs*. This design provides spectral guarantees to preserve connectivity and information flow under aggressive compression, improving the fidelity of the compressed cache without retraining. Our preliminary experiments on GSM8k, AQuA, BBH and LongBench benchmarks show that GEAR-X can achieve competitive or improved accuracy compared to standard GEAR, while maintaining significant memory savings.

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

1 Introduction

Scaling Large Reasoning Models (LRMs) efficiently is increasingly critical as deployment moves into resource-constrained and real-time environments. A critical optimization during inference is *Key-Value (KV) caching*, which stores the intermediate attention states of previously processed tokens. Reusing these states reduces the per-token computational cost from $O(n^2)$ to $O(n)$, where n is the total sequence length. While caching enables fast generation, the cache itself grows linearly with sequence length, creating severe memory bottlenecks that limit throughput and prevent practical deployment in long-context tasks such as chain-of-thought reasoning, code generation, and document-level question answering.

In order to address this, prior work compresses KV caches using techniques like *quantization* [Liu et al., 2024b, Sheng et al., 2023], *low rank* representations [Chang et al., 2025, Lin et al., 2025], *layer-wise* and *head-wise compression* [Liu et al., 2024a, Ge et al., 2024], and *pruning/eviction* [Xiao et al., 2024, Zhang et al., 2023]. Hybrid frameworks such as GEAR [Kang et al., 2024] combine these strategies, storing most entries in ultra-low precision, recovering residuals with a low-rank matrix, and preserving outliers through sparse masks. However, the sparse component in GEAR typically relies on magnitude-based pruning, which can discard structurally important connections and potentially degrade inference quality.

In this work, we propose replacing the unstructured magnitude pruning with *structured sparsity via expander graphs*, which ensure each channel and token maintains multiple well-distributed connections across the network. Expander-based sparsity offers strong spectral guarantees, improving connectivity and preserving informative patterns even under aggressive compression.

2 Related Work

KV cache compression: Reducing the memory footprint of the Key-Value (KV) cache has attracted significant attention as context length scales in LLM inference. *Quantization*-based methods reduce memory by storing cache tensors in low-bit formats. KIVI [Liu et al., 2024b] applies tuning-free 2-bit quantization with asymmetric treatment of keys/values. FLEXGEN [Sheng et al., 2023] formulates tensor placement as a linear programming problem, while KVTUNER [Li et al., 2025] searches for optimal precisions per layer. *Pruning and eviction* approaches discard less important tokens to maintain bounded cache sizes. STREAMINGLLM [Xiao et al., 2024] and H2O [Zhang et al., 2023] evict stale tokens, while TREEKV [He et al., 2025] and SNAPKV [Li et al., 2024] score importance via distance or attention statistics. SEPLLM [Chen et al., 2025] compresses between separators, and FASTGEN [Ge et al., 2024] profiles heads for adaptive eviction.

KV states often admit compact bases. *Low-rank approximations* (PALU [Chang et al., 2025], MATRYOSHKAKV [Lin et al., 2025]) down-project hidden dimensions; LOKI [Singhania et al., 2024] scores tokens in a low-dimensional space. *Sparse representations* like dictionary-based methods (CSR [Zhang et al., 2025], LEXICO [Kim et al., 2025]) achieve sparsity via learned or universal codebooks. Finally, *hybrid frameworks* combine multiple strategies. GEAR [Kang et al., 2024] integrates quantization, sparse outliers, and low-rank correction. LEANKV [Zhang et al., 2024], ROCKETKV [Behnam et al., 2025] mix eviction, sparse attention, and quantization. While hybrids achieve stronger trade-offs, they often lack proper guarantees and rely on heuristic budget allocations.

Structured sparsity: Most KV cache methods rely on magnitude pruning, which prioritizes extreme values but can overlook structurally important entries. Prior work in pruning and compression has shown that structured sparsity often yields better accuracy and hardware efficiency compared to unstructured pruning, due to its more balanced and regular coverage patterns [Wen et al., 2016, Evci et al., 2020].

Expander graphs are particularly attractive: they preserve connectivity under extreme sparsity, supported by well-established spectral guarantees [Marcus et al., 2015, Hoory et al., 2006]. Recent work such as XoRA [Amaljith et al., 2025] has demonstrated their utility for efficient LLM finetuning, though their application to *KV cache compression* remains unexplored.

Our Contribution: Our method extends the GEAR framework, replacing the heuristic magnitude-based outlier selection with expander-driven structured sparsity. This ensures uniform information flow across channels and tokens, thereby reducing the corrective burden on the low-rank component.

Unlike prior hybrid approaches, which rely on ad-hoc sparsity patterns, our framework introduces a theoretically grounded mechanism for sparse selection that remains stable even for long contexts.

3 Our Approach

We build upon the **GEAR** framework [Kang et al., 2024], which achieves KV-cache compression by decomposing each attention key and value matrix into three components: (i) a quantized matrix capturing the bulk of entries, (ii) a sparse term storing extreme-magnitude outliers, and (iii) a low-rank residual approximation to correct systematic quantization errors. This hybrid design enables aggressive compression while eroding accuracy across diverse generation tasks. *Magnitude-based pruning*, which retains both the largest and smallest entries, was adopted to improve quantization precision by isolating extreme values. While simple, this unstructured method can lead to uneven coverage, where certain regions are disproportionately represented, reducing the effectiveness of the sparse backbone.

3.1 Structured Sparsity via Expanders

We replace the magnitude-based sparse selection in GEAR with a *structured sparsity mask* derived from expander graph constructions. This ensures that each channel and token retains multiple, well-distributed connections, preventing isolated or clustered supports.

An *expander graph* is a sparse graph with strong connectivity: every small set of vertices has many edges leading outside the set, thereby preserving robust information flow even under heavy sparsity. We model the KV cache as a bipartite graph connecting channels and token positions, which allows us to analyze sparsity patterns via spectral properties.

For a (d_1, d_2)-biregular bipartite expander, the associated symmetric adjacency has eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{m+n}$, where $\lambda_1 = \sqrt{d_1 d_2}$ and $\lambda_{m+n} = -\sqrt{d_1 d_2}$. The second largest eigenvalue λ_2 governs expansion and mixing behavior, and high *spectral gap* $\lambda_1 - \lambda_2$ guarantees that subsets of rows or columns always connect to many distinct neighbors. By choosing masks from near-optimal expander families like *Ramanujan graphs*, which achieve the bound [Hoory et al., 2006, Marcus et al., 2015]:

$$\lambda_2 \;\leq\; \sqrt{d_1 - 1} + \sqrt{d_2 - 1},$$

Given this condition, we maximize connectivity for the given sparsity budget. For comparison, we also analyzed the spectral gap of magnitude-pruned matrices, and report the results in Section 4.5.

We highlight three properties of expanders that are directly relevant to KV cache compression:

(i) **Uniform retention**: Ensures no subset of rows (channels) or columns (tokens) is drastically under- or over-represented, preventing isolated cache entries that the low-rank correction cannot recover.

(ii) **Spectral contraction**: Guarantees that perturbations introduced by masking or quantization cannot be amplified, yielding a provable upper bound on the residual operator norm.

(iii) **Rapid mixing**: Causes local errors to disperse quickly across rows and columns, flattening the residual spectrum and making it more amenable to low-rank approximation.

Together, these properties ensure that expander-based sparsity maintains balanced support, limits error propagation, and improves the effectiveness of low-rank correction.

3.2 Inference Pipeline

Let $X \in \mathbb{R}^{m \times n}$ be a single KV-cache slice with rows indexing channel coordinates and columns indexing token positions. We write the elementwise (Hadamard) product as $\odot$. Let $Q(\cdot)$ denote the quantizer used for low-bit compression.

Following GEAR [Kang et al., 2024] we decompose X as

$$X \;=\; X_{\text{sparse}} \;+\; \widetilde{X}_{\text{quant}} \;+\; R,$$

where the components are produced by a binary **expander mask** $E \in \{0, 1\}^{m \times n}$:

$$X_{\text{sparse}} \;=\; X \odot E, \qquad \widetilde{X}_{\text{quant}} \;=\; Q\big(X - X_{\text{sparse}}\big).$$

The residual R is then corrected by a rank-r approximation L_r (computed via truncated SVD / power iteration algorithm [Vogels et al., 2019]), producing the reconstruction

$$\widehat{X} \ = \ X_{\text{sparse}} + \widetilde{X}_{\text{quant}} + L_r.$$

3.3 Expander Generation Method

We construct biregular bipartite expander graphs whose biadjacency matrices define the structured sparsity masks. Each token node connects to a fixed number of feature-channel nodes, enforcing uniform degree and preserving information flow across the network. The generation begins by randomly pairing connection "stubs" on both sides to form an initial biregular graph that may contain duplicate or disconnected edges. To efficiently refine this graph while maintaining strict degree constraints, we perform double-edge swaps—local rewiring operations that remove duplicates and incrementally merge disconnected components. This ensures global connectivity and spectral expansion properties without requiring expensive regeneration. To ensure strong expansion quality, we also evaluate the spectral gap of the resulting graph; if it is below a desired threshold, the generation process is repeated until a sufficiently large gap is obtained. In practice, this converges quickly, as random biregular graphs are expanders with high probability.

The resulting biadjacency matrices are stored in compressed sparse row (CSR) format (.npz), minimizing memory and read overhead. On-the-fly generation completes in roughly 3-5 seconds, and the most recently used expanders are cached in memory to further reduce access latency. Together, these optimizations allow structurally guaranteed expanders to be applied with effectively zero additional runtime cost.

4 Experiments

We evaluate our method on four reasoning and long-context benchmarks: GSM8k [Cobbe et al., 2021] (grade-school math word problems), AQuA [Ling et al., 2017] (algebraic word problems with multiple-choice answers), BBH (Big-Bench Hard) [Suzgun et al., 2023] comprising of 23 tasks, and LongBench [Bai et al., 2024] (long-context understanding and reasoning). These tasks were chosen because they require multi-step reasoning or long-context processing and are widely adopted in evaluating compression and quantization methods for large language models.

All experiments on GSM8k, AQuA, and BBH are conducted on the LLaMA-3 8B model [AI, 2024] with 8-shot chain-of-thought prompting [Wei et al., 2022]. LongBench experiments are run on the LLaMA-2 7B model [Touvron et al., 2023]. All runs use a single NVIDIA L4 GPU (24 GB memory), with GSM8k and AQuA completing within a few GPU-hours, while BBH and LongBench required substantially longer (tens of GPU-hours).

Streaming vs. Stream Grouping: We consider two processing settings. In *streaming* (SR), the entire prefix prompt is processed at once, and the decoding phase is quantized in groups of size determined by the streaming gap n_b. In contrast, *stream grouping* (SG) also partitions the prefix: it first processes the largest multiple of n_b as a single quantized block, and any remainder is placed in a residual group (length n_b). During decoding, new tokens first fill this residual group, after which the process continues in blocks of size n_b. Thus, unlike SR, SG ensures that both the prefix and the decoding are aligned with the streaming gap, which reduces mask resizing overhead and yields a more efficient pipeline. Reproduced results show that while SG slightly lags behind SR, incorporating expanders effectively closes this gap (see Section 4.4 for details).

4.1 Results on Reasoning Benchmarks

Table 1 compares our approach with existing methods, all using 4-bit quantization except the FP16 reference. For GEAR-style methods, we follow the same settings: rank $r = 4$ during the prefill phase and rank $r = 2$ for each group of n_b new tokens during decoding, where the streaming gap $n_b = 64$. GEAR and GEAR-L (the latter skips the sparse component) adopt streaming with a sparsity ratio $s = 2\%$. GEAR-X instead uses stream grouping and expander masks with the minimum degree that satisfies regularity: for prefix prompts this corresponds to $s = 1.56\%$, and for decoding $s = 3.12\%$. The originally published and reproduced results for GEAR are compared in Appendix **??**. Detailed performance on all 23 BBH subsets is reported in Table 2.

Table 1: Accuracy results on reasoning benchmarks. Higher is better. Baseline results are reported from Kang et al. [2024]. Results with an asterisk (*) are reproduced by us.

Method	**KV Size**	**GSM8k**	**AQuA**	**BBH**
FP16 (16-bit)	100%	54.21	38.19	53.66
Per-token Quant	34.2%	37.07	**39.37**	46.42
KIVI	34.2%	46.25	36.22	48.03
GEAR-L	29.0%	53.44	38.98	52.23
GEAR	31.0%	54.89*	38.58*	52.74
GEAR-X (Ours)	32.1%	**55.04**	35.04	**53.45**

Overall, our method achieves the best accuracy over GSM8k and BBH while maintaining a comparable compression ratio. The results on BBH highlight substantial variation across tasks: performance is strong on domains such as *web_of_lies*, *sports_understanding*, and *movie_recommendation*, but remains challenging for tasks like *tracking_shuffled_objects_seven_objects* and *dyck_languages*. On AQuA, the performance is slightly lower, but as we show below, this gap can be mitigated by tuning the streaming gap.

Table 2: Performance of GEAR-X on individual BBH tasks.

Subset	**Accuracy**	**Description**
temporal_sequences	0.6960	Reasoning about sequences over time
disambiguation_qa	0.4720	Resolving ambiguous questions
date_understanding	0.7640	Interpreting dates correctly
tracking_shuffled_objects_three_objects	0.5480	Tracking positions of 3 objects
penguins_in_a_table	0.7123	Counting objects in a table
geometric_shapes	0.0960	Identifying shapes and patterns
snarks	0.6011	Logical reasoning puzzles
ruin_names	0.6880	Associating names with ruins
tracking_shuffled_objects_seven_objects	0.0000	Tracking positions of 7 objects
tracking_shuffled_objects_five_objects	0.2960	Tracking positions of 5 objects
logical_deduction_three_objects	0.7640	Deduction with 3 objects
hyperbaton	0.1960	Sentence structure manipulation
logical_deduction_five_objects	0.4560	Deduction with 5 objects
logical_deduction_seven_objects	0.3040	Deduction with 7 objects
movie_recommendation	0.8920	Recommending movies based on preferences
salient_translation_error_detection	0.5480	Detecting errors in translations
reasoning_about_colored_objects	0.7360	Logical reasoning with colored objects
multistep_arithmetic_two	0.1640	Multi-step arithmetic problems
navigate	0.8840	Navigation and spatial reasoning
dyck_languages	0.0920	Recognizing balanced brackets/language patterns
word_sorting	0.2360	Sorting words according to rules
sports_understanding	0.9560	Understanding sports-related scenarios
boolean_expressions	0.8640	Evaluating Boolean logic expressions
object_counting	0.8280	Counting objects in a scene
formal_fallacies	0.1560	Detecting logical fallacies
causal_judgement	0.4813	Inferring causal relationships
web_of_lies	1.0000	Detecting deception in web content

4.2 Results on LongBench

To evaluate long-context performance, we test our method on the LongBench benchmark using the LLaMA-2 7B model. In these experiments, GEAR-X employs expander masks with degrees corresponding to a sparsity ratio of 3.12%, ensuring consistent connectivity across feature channels. The rest of the experimental settings are identical to those described in Section 4.1. As shown in

Table 3, GEAR-X maintains competitive accuracy across diverse long-context tasks while achieving significant KV compression similar to GEAR.

Table 3: Results on LongBench benchmark (LLaMA-2 7B). Higher is better. Fill in values for each dataset and the overall average.

Method	KV Size	NarrQA	Qasper	MFQA-en	MFQA-zh	HotpotQA	2WikiMQA
FP16	100%	17.30	9.08	22.37	19.33	8.24	10.00
GEAR	31.0%	17.30	9.29	22.19	19.13	8.27	10.1
GEAR-X	33.4%	17.32	9.30	22.22	19.09	8.27	10.10

Method	DuReader	GovReport	QMSum	MultiNews	VCSum	TREC	TriviaQA
FP16	23.16	26.76	20.66	5.82	9.91	63.00	84.92
GEAR	23.14	26.99	20.75	5.21	9.91	63.00	84.92
GEAR-X	22.88	27.16	20.80	5.67	9.96	63.00	84.92

Method	SAMSum	LSHT	PCount	PR-en	PR-zh	LCC	RepoBench	MuSiQue	Avg.
FP16	41.44	20.25	1.50	5.77	8.00	58.70	62.30	4.27	24.90
GEAR	41.42	20.25	1.50	5.52	8.00	56.56	60.22	4.26	24.66
GEAR-X	41.33	20.25	1.50	5.52	8.00	56.56	60.22	4.26	24.68

4.3 Impact of Streaming Gap

The streaming gap controls how often the KV Cache is quantized and processed. We vary the gap under our expander-based stream grouping and as shown in Table 4, accuracy peaks at gap size 96, suggesting a trade-off between information freshness and update overhead. For consistency with prior work, we report results with a gap of 64 above, but note that tuning the gap provides an avenue for further improvement, particularly on AQuA.

Table 4: Effect of streaming gap size (GEAR-X).

Gap Size	GSM8k	AQuA
64	55.04	35.04
96	**55.26**	**37.01**
128	54.89	36.61

4.4 Effects of Replication and Streaming Settings

In Table 5, we report the results of reproducing streaming (SR) and stream grouping (SG) baselines on our system. Unless otherwise stated, all parameters and settings are the same as those described in Section 4.1. On GSM8k and AQuA, we observe that SG slightly lags behind SR, but our expander-based SG effectively closes this gap. The discrepancy observed between the originally published and reproduced numbers is due to differences in hardware and implementation. These results suggest that expanders hold promise for improving accuracy in the streaming setting, which we intend to explore further in future work.

Table 5: Reproduced results of streaming (SR) and stream grouping (SG).

Method	Setting	GSM8k	AQuA
GEAR (Published)	SR	54.76	40.55
GEAR (Reproduced)	SR	54.89	38.58
GEAR (Reproduced)	SG	54.43	37.40
GEAR-X	SG	**55.04**	35.04

4.5 Spectral Gap Analysis of Magnitude-Pruned Matrices

We evaluated the spectral gaps of the sparse matrices produced by *magnitude pruning* (using the same eigen-operator and convention as in the main text). Across layers and settings, the observed gaps are uniformly small—typically $\lambda_1 - \lambda_2 \lesssim 5$ (often in the 1–4 range), in contrast to our constructed expanders, which exhibit much larger gaps (often 20–30). These results indicate that magnitude-pruned matrices do *not* exhibit expander behavior, consistent with our initial assumptions and rationale.

5 Conclusion

Our study demonstrates that expander graphs provide a principled and robust backbone for KV cache compression. Their spectral guarantees confer structural advantages over unstructured magnitude pruning, and our experiments show that this approach achieves competitive or improved accuracy while maintaining significant memory savings.

While our current evaluation is limited by computational constraints, we plan on extending it to include detailed efficiency reporting in terms of throughput (tokens/s), latency, and peak memory usage. Beyond this, we also aim to investigate expanders more extensively for KV cache compression, including their standalone effect, independent of GEAR. We will further explore more efficient streaming strategies, and study their integration with complementary compression methods like structured projections.

Overall, our work reinforces the view that expanders are a useful and theoretically grounded tool for advancing efficient inference, addressing key constraints of memory, latency, and scalability.

References

Meta AI. Introducing Meta Llama 3: The most capable openly available LLM to date. `https://ai.meta.com/blog/meta-llama-3/`, 2024.

E. V. Amaljith, Arindam Biswas, Suryam Arnav Kalra, Pabitra Mitra, and Biswajit Basu. Xora: Expander adapted lora finetuning. In *ICLR 2025 Workshop—Submitted*, 2025. URL `https://openreview.net/forum?id=uBnVA7SeWv`. Under review.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL `https://aclanthology.org/2024.acl-long.172`.

Payman Behnam, Yaosheng Fu, Ritchie Zhao, Po-An Tsai, Zhiding Yu, and Alexey Tumanov. RocketKV: Accelerating long-context LLM inference via two-stage KV cache compression. In *Forty-second International Conference on Machine Learning*, 2025. URL `https://openreview.net/forum?id=RyOpooIxDF`.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed S. Abdelfattah, and Kai-Chiang Wu. Palu: KV-cache compression with low-rank projection. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=LWMS4pk2vK`.

Guoxuan Chen, Han Shi, Jiawei Li, Yihang Gao, Xiaozhe Ren, Yimeng Chen, Xin Jiang, Zhenguo Li, Weiyang Liu, and Chao Huang. SepLLM: Accelerate large language models by compressing one segment into one separator. In *Forty-second International Conference on Machine Learning*, 2025. URL `https://openreview.net/forum?id=MhVJCxYEEi`.

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

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2943–2952. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/evci20a.html`.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive KV cache compression for LLMs. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=uNrFpDPMyo.

Ziwei He, Jian Yuan, Haoli Bai, Jingwen Leng, and Bo Jiang. Treekv: Smooth key-value cache compression with tree structures. *CoRR*, abs/2501.04987, January 2025. URL https://doi.org/10.48550/arXiv.2501.04987.

Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006.

Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. GEAR: An efficient error reduction framework for KV cache compression in LLM inference. In *Proceedings of The 4th NeurIPS Efficient Natural Language and Speech Processing Workshop*, volume 262 of *Proceedings of Machine Learning Research*, pages 305–321. PMLR, 14 Dec 2024. URL https://proceedings.mlr.press/v262/kang24a.html.

Junhyuck Kim, Jongho Park, Jaewoong Cho, and Dimitris Papailiopoulos. Lexico: Extreme KV cache compression via sparse coding over universal dictionaries. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=Yh9vxlxnjA.

Xing Li, Zeyu XING, Yiming Li, Linping Qu, Hui-Ling Zhen, Yiwu Yao, Wulong Liu, Sinno Jialin Pan, and Mingxuan Yuan. KVTuner: Sensitivity-aware layer-wise mixed-precision KV cache quantization for efficient and nearly lossless LLM inference. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=zDwipF6h06.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=poE54GOq2l.

Bokai Lin, Zihao Zeng, Zipeng Xiao, Siqi Kou, TianQi Hou, Xiaofeng Gao, Hao Zhang, and Zhijie Deng. MatryoshkaKV: Adaptive KV compression via trainable orthogonal projection. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=BQwsRy1h3U.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1015. URL https://aclanthology.org/P17-1015/.

Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. Minicache: KV cache compression in depth dimension for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=sgVOjDqUMT.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Forty-first International Conference on Machine Learning*, 2024b. URL https://openreview.net/forum?id=L057s2Rq8O.

Adam W Marcus, Daniel A Spielman, and Nikhil Srivastava. Interlacing families I: Bipartite ramanujan graphs of all degrees. *Ann. Math.*, 182(1):307–325, 2015.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Re, Ion Stoica, and Ce Zhang. FlexGen: High-throughput generative inference of large language models with a single GPU. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 31094–31116. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/sheng23a.html.

Prajwal Singhania, Siddharth Singh, Shwai He, Soheil Feizi, and Abhinav Bhatele. Loki: Low-rank keys for efficient sparse attention. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=raABeiV71j.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824/.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL `https://arxiv.org/abs/2307.09288`.

Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. Powersgd: Practical low-rank gradient compression for distributed optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper_files/paper/2019/file/d9fbed9da256e344c1fa46bb46c34c5f-Paper.pdf`.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=_VjQlMeSB_J`.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL `https://proceedings.neurips.cc/paper_files/paper/2016/file/41bfd20a38bb1b0bec75acf0845530a7-Paper.pdf`.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=NG7sS51zVF`.

Hongxuan Zhang, Yao Zhao, Jiaqi Zheng, Chenyi Zhuang, Jinjie Gu, and Guihai Chen. Csr:achieving 1 bit key-value cache via sparse representation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39 (24):25860–25867, 04 2025. doi: 10.1609/aaai.v39i24.34779. URL `https://ojs.aaai.org/index.php/AAAI/article/view/34779`.

Yanqi Zhang, Yuwei Hu, Runyuan Zhao, John C. S. Lui, and Haibo Chen. Unifying kv cache compression for large language models with leankv. *CoRR*, abs/2412.03131, 2024. URL `https://doi.org/10.48550/arXiv.2412.03131`.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Re, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=RkRrPp7GKO`.