

E8 Lattice Key Quantization for Low-Bit KV-Cache Compression

Sasho Nedelkoski

Technical University of Berlin
Berlin, Germany

sasho.nedelkoski@campus.tu-berlin.de

Alexander Acker

logsight.ai

Berlin, Germany

alexander.acker@logsight.ai

Soeren Becker

logsight.ai

Berlin, Germany

soeren.becker@logsight.ai

Dominik Scheinert

logsight.ai

Berlin, Germany

dominik.scheinert@logsight.ai

Odej Kao

Technical University of Berlin
Berlin, Germany

odej.kao@tu-berlin.de

Abstract

Compressing the key-value (KV) cache is essential for serving long-context large language models (LLMs) under memory constraints. While current scalar quantization methods achieve significant compression, they typically face a quality-fidelity trade-off, either suffering from performance instability at 2-bit precision or necessitating 3-bit keys to maintain model quality. In this work, we demonstrate that a uniform low-bit k2v2 configuration (2-bit keys and 2-bit values) can preserve near-baseline quality by using vector quantization (VQ) based on the E_8 lattice for KV keys. We show that structured VQ exploits correlations across adjacent dimensions to achieve signal-to-quantization-noise ratios (SQNR) higher than those of independent scalar quantization at the same key payload rate. To ensure practical utility, we adapt a table-based asymmetric scoring algorithm that avoids explicit codebook dequantization during attention computation and preserves decode-throughput parity with scalar baselines. Across five LLM families, our E_8 k2v2 approach yields $\sim 6\times$ KV compression, roughly 20% smaller than scalar k3v2 (3-bit keys and 2-bit values), while retaining near-baseline long-context perplexity. In our full-suite perplexity evaluation, E_8 improves over scalar k3v2 in 80% of model–dataset–context settings. These results indicate that E_8 lattice VQ improves quality per byte while keeping throughput at parity, suggesting that structured key quantization is a useful primitive for KV-compression stacks. Overall, our work provides a low-overhead primitive for expanding the quality-memory frontier in long-context LLM inference.

1 Introduction

Supporting long context windows in large language models (LLMs) is essential for in-context learning, long-horizon reasoning, holistic codebase analysis, and retrieval-augmented generation [Wang et al., 2024]. LLM inference requires storing key-value (KV) pairs for every token in the context window

across all attention layers and heads, placing high pressure on available device memory [Li et al., 2025a]. For a model with L layers, H_{kv} KV heads, and head dimension d , the KV cache consumes $2 \cdot L \cdot H_{kv} \cdot d \cdot 2$ bytes per token in fp16. At 128K tokens of context, even an 8B grouped-query model requires tens of gigabytes of KV cache, and models with more KV heads can exceed the memory footprint of the model weights themselves. This bottleneck limits batch size and context length, leading to prohibitive deployment cost. Consequently, KV-cache efficiency directly affects whether long-context serving fits on a single accelerator or requires multi-GPU execution.

To mitigate these memory constraints, KV cache quantization has emerged as a prominent solution [Li et al., 2025a]. Recent KV-cache quantization approaches, including KIVI [Liu et al., 2024], KVQuant [Hooper et al., 2024], and TurboQuant [Zandieh et al., 2025], rely primarily on per-channel scalar quantization. Treating each of the key dimensions d as an independent signal, a quantization of b bits assigns each dimension to one of the 2^b levels, achieving a rate of b bits/dim. However, scalar quantization often incurs quality degradation at very low bitrates. Recent hardware-optimized configurations typically necessitate 3 bits for keys (e.g., TurboQuant’s k3v2 setting: 3-bit keys and 2-bit values) to preserve model quality.

We hypothesize that the degradation of low-bit scalar key quantization is partly due to limited use of geometric structure in key vectors. While scalar methods are computationally efficient, they treat each dimension in isolation, ignoring the inter-dimensional correlations prevalent in LLM hidden states [Engels et al., 2025]. To capture this structure, we use lattice VQ over blocks of dimensions to exploit space-filling and shaping gains, which can provide higher SQNR than scalar quantization at matched bit rates [Zamir and Feder, 1996].

Specifically, we employ the E_8 lattice in $\mathbb{R}^8$, which offers a useful combination of strong lattice geometry and practical efficiency. It achieves the densest possible sphere packing in eight dimensions [Viazovska, 2017], with near-spherical Voronoi cells that provide a shaping advantage over cubic scalar quantizers. Our implementation uses a structured E_8 -derived (E8P12-style) indexing scheme with parity/sign constraints and a two-branch ± 0.25 selection rule, enabling efficient nearest-codeword assignment without brute-force search over all codewords.

Contributions. This paper makes the following contributions:

1. We apply E_8 lattice VQ to key compression in the KV cache, partitioning each d -dimensional key vector into independent 8-dimensional blocks. At 2 bits/dim payload (65,536 codewords per block; 2.25 bits/dim effective key rate including norm metadata), the resulting k2v2 configuration achieves $\sim 6\times$ total KV compression relative to fp16. Reported memory numbers are end-to-end KV-cache bytes/token, counting keys, values, quantized indices/signs, norms, and value scale/zero metadata.
2. We show that E_8 at low key rate achieves rate-distortion performance between scalar 3-bit and 4-bit quantization. On 128-dimensional Gaussian vectors, E_8 obtains 10.47 dB SQNR, compared with 7.37 dB for scalar 3-bit, while using a lower effective key rate. In a controlled attention-fidelity probe, the 2-bit E_8 configuration also yields lower average attention KL divergence than scalar k3v2 (0.045 vs. 0.091) and KIVI k2v2 (0.224), indicating better preservation of fp16 attention distributions under compression.
3. To avoid dequantizing vector-quantized keys during decoding, we adapt table-based asymmetric scoring to the fixed E_8 key codebook. For each query-head block, we precompute the dot product between the query block and all 65,536 E_8 codewords. Each compressed key block can then be scored by indexing this table with its stored codeword id, replacing an 8-dimensional dot product with a lookup. This removes explicit key dequantization from the decode-time scoring path in our serving path and matches the throughput of optimized scalar decoders in the measured setting.
4. We provide full-suite evaluations across five LLMs (Qwen2.5, Llama-3.1, Mistral, Falcon3, and Granite). The results show that E_8 k2v2 improves the quality–memory trade-off over scalar k3v2 in most long-context perplexity settings and has lower perplexity than KIVI k2v2 across the evaluated perplexity suite, without a measurable decode-throughput penalty.

2 Related work

KV-cache quantization. Current KV-cache compression methods primarily use scalar quantization. KIVI [Liu et al., 2024] uses per-channel asymmetric key quantization and per-token value quantization; KVQuant [Hooper et al., 2024] adds pre-RoPE keys, non-uniform calibrated grids, and sparse outlier handling; GEAR [Kang et al., 2024] combines low-rank approximation with quantized residuals; TurboQuant [Zandieh et al., 2025] uses QJL random projections for calibration-free robustness, and its k3v2 setting is our primary scalar baseline. ZipCache [He et al., 2024], LeanKV [Zhang et al., 2025b], and AQUA-KV [Shutova et al., 2025] further adapt bit-widths across tokens, keys/values, or predictable residual components. These methods are effective, but the dominant coding operation remains scalar or projection-based; our work instead encodes each key block with a structured geometry-aware codebook.

Joint coding and vector quantization. Coupled Quantization (CQ) [Zhang et al., 2024] observes that joint KV-channel entropy can be far below the sum of marginal entropies, motivating cross-channel coding; PM-KVQ [Liu et al., 2025] highlights the need for long-context-aware calibration under RoPE. Recent VQ-based KV-cache systems, including NestQuant [Savkin et al., 2025] and CommVQ [Li et al., 2025b], further show that lattice or learned vector codebooks can improve low-bit cache compression. Our method shares the premise that key channels should not be treated independently, but differs by using a fixed, non-learned E_8 key codebook with online encoding and by adding decode-time table scoring so attention can operate directly on compressed keys. Related VQ work for weights, including PQ [Jegou et al., 2011, Guo et al., 2020], QuIP [Chee et al., 2023], QuIP# [Tseng et al., 2024], and AQLM [Egiazarian et al., 2024], shows that structured codebooks can achieve strong low-bit rate-distortion. KV caches impose different constraints: activations are produced online during decoding and attention scores must be computed against a growing cache, so avoiding dense key reconstruction is central.

Token reduction and attention approximation. Token-eviction systems reduce cache length rather than bits per token. StreamingLLM [Xiao et al., 2024] retains recent tokens and attention sinks; H2O [Zhang et al., 2023] and SnapKV [Li et al., 2024] retain tokens selected by accumulated or pooled attention; MagicPIG [Chen et al., 2024] uses LSH and CPU-resident hash tables for sparse long-context attention. Projection methods such as Loki [Singhania et al., 2024] reduce effective key dimensionality before attention. These approaches are orthogonal to our quantizer: one could apply E_8 key coding to the retained token set or to a projected subspace.

Lattice codes. Lattice quantization theory [Zamir and Feder, 1996, Conway and Sloane, 1999] shows that structured lattices achieve space-filling gain that scalar quantizers cannot. The E_8 lattice is the densest known sphere packing in eight dimensions [Viazovska, 2017]; its 240 kissing neighbors and near-spherical Voronoi cells give a shaping advantage over cubic lattices and motivate its use in low-bit neural compression. To our knowledge, this is the first application of E_8 coding to runtime KV caches with online encoding and throughput-neutral compressed-domain attention scoring.

3 Method

3.1 Preliminary baseline

As a point of comparison, standard scalar KV-cache quantization independently quantizes each key dimension. Consider a key vector $\mathbf{k} \in \mathbb{R}^d$, where d is the attention head dimension. Scalar quantization at b bits independently quantizes each dimension:

$$\hat{k}_i = s \cdot \text{round}\left(\frac{k_i}{s}\right), \quad s = \frac{\max_i |k_i|}{2^{b-1} - 1}, \quad (1)$$

storing $b \cdot d$ key-index bits plus scale metadata. Eq. (1) is a generic per-vector scalar-quantization reference used to introduce notation; concrete KV baselines may use per-channel, per-token, or groupwise scaling. Our implemented scalar baseline (TurboQuant k3v2) uses rotated unit-norm keys with a Lloyd-Max MSE stage plus a 1-bit QJL residual-sign stage. Throughout the paper, we assume that d is divisible by 8, and as our running example we choose $d = 128$ to make the explanation clear. However, the framework generalizes to arbitrary head dimensions satisfying this constraint.

Compression ratios refer to *effective KV-cache bytes per token*, not nominal key bitwidth alone. A complete fp16 KV cache stores one key and one value vector per token-head, requiring $2 \cdot d \cdot 16$ bits. As an example, for $d = 128$, this is 4096 bits per token-head. As an example, the TurboQuant k3v2 [Zandieh et al., 2025] setting stores $3d$ packed key bits and $2d$ packed value bits, plus metadata for dequantization and compressed-domain scoring:

$$B_{S3} = 3d + 2d + m_{S3}.$$

End-to-end accounting includes packed keys, packed values, and per-token metadata such as QJL sign bits, norms, and value scale/zero parameters; it excludes fixed model-level state such as rotation matrices and shared codebooks. According to this convention, the k3v2 scalar yields about $\sim 4.8\times$ compression over fp16, as shown in Table 2 below.

3.2 E8 lattice vector quantization for keys

The E_8 lattice $\Lambda_{E_8} \subset \mathbb{R}^8$ is defined as

$$\Lambda_{E_8} = \left\{ \mathbf{x} \in \mathbb{Z}^8 \cup (\mathbb{Z} + \frac{1}{2})^8 \mid \sum_{i=1}^8 x_i \equiv 0 \pmod{2} \right\}. \quad (2)$$

The full E_8 lattice underlies the optimal sphere packing in 8D space [Viazovska, 2017], making it a natural high-dimensional quantization template. We use a fixed, non-learned E_8 -derived codebook $\mathcal{C} \subset \mathbb{R}^8$ with $2^{16} = 65,536$ codewords, so each 8-dimensional key block is represented by one 16-bit index. The codebook is built from an E_8 -consistent half-integer grid augmented with norm-12 patterns. Its indexing structure separates an 8-bit absolute-value prototype from an 8-bit sign/parity pattern, enabling deterministic encode/decode without brute-force search over all codewords. During encoding, we evaluate two ± 0.25 shifted lattice branches and store the lower-error index. This work focuses on the fixed E_8 key codec itself and does not add a separate sparse-outlier retention path. Such outlier handling is complementary and left for future work.

Encoding. To quantize a key vector $\mathbf{k} \in \mathbb{R}^d$, the method applies a random orthogonal rotation matrix $\mathbf{R} \in \mathbb{R}^{d \times d}$ (shared across all keys in a layer and fixed at initialization) to spread energy across coordinates and mitigate channel-aligned outliers while preserving L_2 distances. Following the assumption that d is a multiple of 8 (which is the case in most modern LLMs [Wang et al., 2025]), we then partition the rotated vector into $M = \frac{d}{8}$ discrete 8-dimensional blocks. For $d = 128$, this gives $M = 16$ blocks:

$$\tilde{\mathbf{k}} = \mathbf{R}\mathbf{k}, \quad \tilde{\mathbf{k}} = [\tilde{\mathbf{k}}^{(1)}, \dots, \tilde{\mathbf{k}}^{(16)}], \quad \tilde{\mathbf{k}}^{(j)} \in \mathbb{R}^8. \quad (3)$$

Each block is scaled and quantized to its nearest E_8 codeword:

$$z_j = \arg \min_{i \in \{0, \dots, |\mathcal{C}|-1\}} \left\| \frac{\alpha}{\|\tilde{\mathbf{k}}\|_2} \tilde{\mathbf{k}}^{(j)} - \mathbf{g}_i \right\|_2^2, \quad j = 1, \dots, 16, \quad (4)$$

where α is a calibrated scale parameter. In all main experiments, α is selected deterministically by minimizing block reconstruction MSE on 300 seeded Gaussian unit vectors using a coarse scale grid followed by local refinement; for $d = 128$ this gives $\alpha \approx 12$. The compressed key is stored as 16×16 index bits plus one 32-bit norm $\|\tilde{\mathbf{k}}\|_2$, totaling $16 \times 16 + 32 = 288$ bits = 36 bytes, yielding a rate of $288/128 = 2.25$ bits/dim.

Effective rate. For $d = 128$, the raw packed E_8 k2v2 payload is 288 key bits plus $2d = 256$ value bits, or 544 bits per token-head before value metadata. Under the accounting convention above, this gives the $\sim 6\times$ end-to-end KV-cache compression over fp16 (Table 2).

3.3 Table-based asymmetric scoring

A naive implementation of attention with lattice VQ keys would dequantize each key by looking up its M codewords from the E_8 grid, reconstructing a dense d -dimensional vector, then computing the dot product with the query. For N cached keys, this requires NM random accesses to a $65,536 \times 8$ float32 table—a memory bandwidth bottleneck that typically renders lattice VQ slower

than scalar quantization. We instead use *table-based asymmetric scoring*, following asymmetric-distance computation in product quantization [Jegou et al., 2011] and recent lookup-based KV-attention systems [Zhang et al., 2025a, Wu et al., 2025, Karmore, 2026]. Given a query $\mathbf{q} \in \mathbb{R}^d$ for a single decoding step:

1. Rotate and partition: We apply the same orthogonal rotation to the query $\tilde{\mathbf{q}} = \mathbf{R}\mathbf{q}$, and partition it into M blocks $\tilde{\mathbf{q}}^{(j)} \in \mathbb{R}^8$ (e.g., $M = 16$ for $d = 128$).
2. Pre-compute lookups: We build M lookup tables: $T_j[i] = \langle \tilde{\mathbf{q}}^{(j)}, \mathbf{g}_i \rangle$ for $i = 0, \dots, 65,535$, where $\mathbf{g}_i$ is the i -th E_8 codeword. Each of the M tables contains 65,536 float32 entries.
3. Asymmetric scoring: For each cached key stored as M indices $(z_1, \dots, z_M)$ and norm $n = \|\tilde{\mathbf{k}}\|_2$, we compute the attention score using the calibrated parameter α from Eq. (4):

$$\text{score} = \frac{n}{\alpha} \sum_{j=1}^M T_j[z_j]. \quad (5)$$

Computational efficiency. Step 2 costs $M \times 65,536$ 8-dimensional dot products, implemented as a batched matrix multiplication over query blocks and codewords. Step 3 costs NM table gathers plus NM additions. The M lookup tables require $M \times 65,536 \times 4$ bytes of memory. For $M = 16$, this totals 4 MB per query head. Because each gather is a single indexed read from a contiguous query-local table, per-key scoring is memory-efficient in our decode setting; only the once-per-query table build scales with codebook size. For batched teacher-forcing quality evaluations (perplexity and downstream tasks), we use the same compressed representation and asymmetric inner-product estimator. These batched evaluations use a mathematically equivalent dense reconstruction when processing many query positions at once; the decode throughput results in Table 3 exercise the table-based serving path directly.

3.4 Implementation details

We integrate E_8 key quantization into the TurboQuant [Zandieh et al., 2025] framework, retaining its value quantization and residual window for recently inserted tokens. The reported E_8 k2 key path does not use QJL residual signs; QJL is retained only in scalar TurboQuant baselines. The random orthogonal rotation matrix $\mathbf{R} \in \mathbb{R}^{d \times d}$ is initialized once per layer from a seeded PRNG. For $d = 128$, storing this dense matrix in fp32 requires 64 KB per layer (or 32 KB in fp16), which is a negligible model-level overhead in our accounting. Encoding a single key requires one matrix-vector multiply followed by M fast structured nearest-codeword searches over 8D blocks. Rather than brute-force $O(|\mathcal{C}|)$ VQ lookup, the implementation first rounds in a reduced absolute-value candidate set, then applies sign/parity-consistent index reconstruction with the ± 0.25 branch selection. As shown in the rate-distortion analysis in Figure 1, this integration yields an E_8 configuration whose SQNR lies between scalar 3-bit and 4-bit baselines while remaining in the low-bit key-rate regime.

4 Experiments

4.1 Setup

Models. We evaluate on five instruction-tuned LLMs spanning different architectures and labs: Qwen2.5-14B [Qwen Team et al., 2025], Mistral-7B [Jiang et al., 2023], Llama-3.1-8B [Grattafiori et al., 2024], Falcon3-7B [Almazrouei et al., 2023], and Granite-3.1-8B [Stallone et al., 2024]. All models use grouped-query attention with head dimension divisible by 8 and varying numbers of KV heads.

Configurations. We compare four KV cache configurations: (1) Baseline: fp16 keys and values; (2) KIVI k2v2 [Liu et al., 2024]; (3) Scalar k3v2 (TurboQuant [Zandieh et al., 2025]); and (4) our E8 Lattice k2v2 setting. All compressed configurations use 2-bit values and a recent-token residual buffer of 128 tokens unless explicitly varied in residual-window sweeps.

Table 1: Perplexity on WikiText-2 and C4 across context lengths. Each block reports compressed-method perplexity deltas against fp16 baseline; lower is better. Bold marks the best compressed configuration within each dataset/model/context.

Model	Ctx	WikiText-2								C4							
		Base	KIVI	Δ	Scalar	Δ	E8	Δ		Base	KIVI	Δ	Scalar	Δ	E8	Δ	
Qwen2.5-14B	512	5.90	7.16	+1.26	6.29	+0.39	6.13	+0.23		11.11	12.47	+1.36	11.59	+0.48	11.35	+0.24	
	2K	5.01	7.32	+2.31	5.61	+0.60	5.29	+0.28		10.32	13.82	+3.50	11.31	+0.99	10.60	+0.29	
	4K	4.84	7.88	+3.04	5.51	+0.68	5.13	+0.29		10.18	15.38	+5.20	11.59	+1.41	10.47	+0.29	
Llama-3.1-8B	512	7.33	9.40	+2.07	7.70	+0.36	7.51	+0.18		11.34	15.00	+3.65	11.93	+0.59	11.71	+0.37	
	2K	6.45	15.91	+9.45	6.97	+0.51	6.67	+0.21		10.90	32.72	+21.82	11.84	+0.94	11.35	+0.45	
	4K	6.30	21.47	+15.17	6.90	+0.60	6.54	+0.23		10.88	47.43	+36.54	12.05	+1.17	11.40	+0.52	
Mistral-7B	512	5.75	6.25	+0.50	5.85	+0.10	5.81	+0.07		8.87	9.51	+0.64	8.99	+0.12	8.93	+0.06	
	2K	4.98	5.93	+0.94	5.12	+0.14	5.06	+0.07		8.21	9.72	+1.51	8.42	+0.21	8.28	+0.07	
	4K	4.84	6.12	+1.27	5.00	+0.16	4.92	+0.08		8.10	10.25	+2.15	8.35	+0.25	8.18	+0.08	
Falcon3-7B	512	8.91	9.40	+0.49	8.87	-0.04	9.08	+0.17		17.85	18.53	+0.68	17.89	+0.04	17.96	+0.11	
	2K	7.19	7.79	+0.60	7.08	-0.11	7.30	+0.11		15.94	17.12	+1.19	15.82	-0.11	15.98	+0.04	
	4K	6.63	7.16	+0.53	6.57	-0.06	6.72	+0.09		15.16	16.67	+1.51	15.19	+0.03	15.24	+0.08	
Granite-3.1-8B	512	5.53	6.68	+1.15	5.90	+0.36	5.69	+0.16		8.49	9.85	+1.36	8.85	+0.36	8.68	+0.19	
	2K	4.97	7.22	+2.25	5.52	+0.55	5.19	+0.22		8.25	11.57	+3.33	8.98	+0.74	8.59	+0.35	
	4K	4.89	8.11	+3.22	5.58	+0.69	5.19	+0.30		8.34	13.14	+4.79	9.28	+0.94	8.85	+0.50	

Metrics. To provide a holistic assessment, we categorize our evaluation into three axes: (1) End-to-end quality: long-context perplexity on WikiText-2 [Merity et al., 2016] and C4 [Dodge et al., 2021] (contexts 512, 2048, 4096), downstream few-shot accuracy on HellaSwag (10-shot) [Zellers et al., 2019], ARC-Challenge (25-shot) [Clark et al., 2018], and WinoGrande (5-shot) [Sakaguchi et al., 2019], alongside a needle-in-a-haystack (NIAH) retrieval sanity check. (2) Internal fidelity: per-layer cosine similarity between compressed and fp16 key representations, and KL divergence for attention distributions. (3) System performance: single-sequence decode throughput (tokens/s) on a single 80 GB NVIDIA A100 GPU and total memory consumption. We report single-system, single-sequence serving measurements; larger-batch scheduling behavior, cross-hardware variance, and cross-backend variance are left to future work.

4.2 Rate-distortion analysis

Figure 1 shows rate-distortion curves for our implemented TurboQuant-style scalar/product key baseline and the E_8 lattice key quantizer on 128-dimensional Gaussian vectors. At the low-rate operating point, E_8 achieves SQNR of 10.47 dB versus 2.43 dB for the 2-bit scalar/product baseline. More practically, E_8 at 2 bits/dim (SQNR 10.47 dB, cosine similarity 0.954) falls *between* the implemented scalar 3-bit (7.37 dB, 0.920) and 4-bit (12.69 dB, 0.974) configurations. This comparison supports the empirical quality-per-bit advantage of the fixed E_8 key codec in our compression stack; it should not be read as an optimal scalar-quantizer lower bound.

4.3 Perplexity

Table 1 shows complete WikiText-2 and C4 perplexity at contexts 512/2048/4096 for all five full-suite models. E_8 k2v2 improves over scalar k3v2 in 24/30 model–dataset–context blocks while using fewer key bits; these wins include every non-Falcon model and both corpora. The gains are most visible at longer context lengths, where quantization error accumulates over more cached keys. For example, at 4K context on C4, Qwen2.5-14B degrades by only +0.29 perplexity under E_8 , compared with +1.41 for scalar k3v2 and +5.20 for KIVI; Llama-3.1-8B shows the same pattern (+0.52 for E_8 , +1.17 for

Figure 1: E_8 Lattice VQ at 2 bits/dim Approaches Scalar VQ at 4 bits/dim

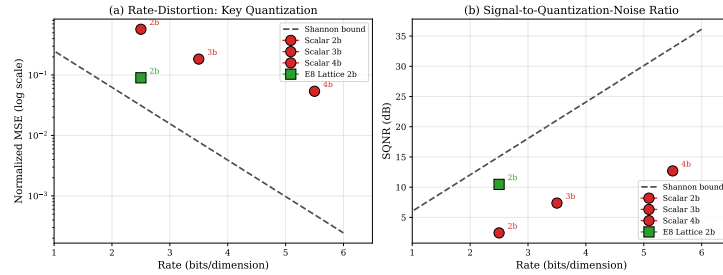


Figure 1: Rate-distortion comparison on 128-dimensional Gaussian vectors. E_8 lattice VQ achieves SQNR between scalar 3-bit and 4-bit quantization at a comparable low key rate.

Table 2: Effective compressed KV-cache bytes per token for the quantized cache region. The accounting includes packed key/value data and per-token metadata (key norms/QJL signs and value scale/zero groups), but excludes fixed model-level tables/rotations and the finite recent-token residual window.

Model	L	H_{kv}	Baseline (B/tok)	Scalar k3v2		E8 Lattice k2v2	
				(B/tok)	Ratio	(B/tok)	Ratio
Qwen2.5-14B	48	8	196,608	40,944	$4.8\times$	32,592	$6.0\times$
Llama-3.1-8B	32	8	131,072	27,296	$4.8\times$	21,728	$6.0\times$
Mistral-7B	32	8	131,072	27,296	$4.8\times$	21,728	$6.0\times$
Falcon3-7B	28	4	114,688	22,652	$5.1\times$	18,452	$6.2\times$
Granite-3.1-8B	40	8	163,840	34,120	$4.8\times$	27,160	$6.0\times$

scalar, +36.54 for KIVI). Granite-3.1-8B also benefits substantially at 4K, with WikiText-2/C4 deltas of +0.30/+0.50 for E_8 versus +0.69/+0.94 for scalar.

The only systematic scalar advantage appears on Falcon3-7B, where scalar k3v2 is best in all six Falcon rows. Even there, the gap is small in absolute perplexity (at most 0.22), and E_8 remains below KIVI in every Falcon row. This suggests that Falcon’s activation statistics are unusually favorable to per-coordinate scalar quantization, but not that the 2-bit lattice representation is generally brittle. Overall, the robust conclusion is that E_8 moves the quality–memory Pareto frontier beyond scalar k3v2 for most evaluated long-context settings and dominates KIVI k2v2 across this perplexity suite.

4.4 Memory compression

Table 2 quantifies per-token KV cache memory. E_8 k2v2 achieves $\sim 6\times$ effective compression for the four $H_{kv} = 8$ models, compared with $\sim 4.8\times$ for scalar k3v2. This corresponds to roughly a 20% smaller compressed cache than scalar k3v2 after accounting for packed key/value payloads and per-token metadata. The Falcon3-7B row is slightly different because its larger head dimension better amortizes fixed per-vector metadata such as key norms, yielding $5.1\times$ for scalar and $6.2\times$ for E_8 .

These ratios use the effective bytes/token accounting defined in Section 3. For a representative 8B architecture such as Llama-3.1-8B, scalar k3v2 requires 27.3 KB/token, whereas E_8 k2v2 requires 21.7 KB/token, saving 5.6 KB per token in the quantized cache region. At 128K context length, this accounting projects roughly 700 MB of additional memory saved per sequence.

4.5 Decode throughput

Table 3 demonstrates that table-based asymmetric scoring (detailed in Section 3.3) successfully eliminates the computational bottleneck traditionally associated with lattice VQ. Across all five evaluated architectures and context lengths, E_8 k2v2 achieves decode-throughput parity with both the uncompressed fp16 baseline and the optimized scalar (k3v2) and KIVI kernels in our single-sequence decode benchmark. The variance across all methods falls within the margin of standard measurement noise (± 0.2 tok/s), indicating no measurable latency degradation in this setting. This empirical parity supports our system design in the measured setting: precomputing the lookup table avoids dense key reconstruction during attention, and the remaining table-build/gather overhead is below measurement noise for our single-sequence A100 decode path. These throughput results are measured on our A100 serving path; additional validation on other accelerators and paged-attention runtimes is left for future work.

4.6 Downstream task accuracy

To ensure that extreme KV cache compression does not degrade foundational model capabilities, we evaluate zero-shot and few-shot reasoning performance on HellaSwag, ARC-Challenge, and WinoGrande. Table 4 reports these downstream accuracies. Across all models, E_8 k2v2 strictly maintains parity with both the fp16 baseline and the scalar baselines. In few-shot settings, much of the demonstration context is still quantized; the observed parity therefore suggests that the compressed

Table 3: Decode throughput (tok/s) with table-based asymmetric scoring. E8 Lattice matches Scalar throughput despite using structured VQ, thanks to lookup-table scoring that avoids codebook dequantization.

Model	Context	Baseline	KIVI k2v2	Scalar k3v2	E8 Lattice k2v2
Qwen2.5-14B	1K	49.4	49.4	49.4	49.4
	4K	48.4	48.4	48.4	48.4
Llama-3.1-8B	1K	89.0	88.9	88.9	89.0
	4K	86.9	87.0	87.0	87.0
Mistral-7B	1K	92.9	92.9	92.9	93.0
	4K	90.7	90.7	90.8	90.8
Falcon3-7B	1K	94.4	94.5	94.5	94.5
	4K	91.2	91.2	91.3	91.2
Granite-3.1-8B	1K	81.0	81.0	81.0	81.0
	4K	78.7	78.7	78.7	78.7

Table 4: Downstream task accuracy (%). Each cell reports HellaSwag 10-shot / ARC-Challenge 25-shot / WinoGrande 5-shot / average.

Model	Baseline	KIVI k2v2	Scalar k3v2	E8 k2v2
Cell format: HellaSwag / ARC-C / WinoGrande / Avg				
Qwen2.5-14B	65.5 / 60.5 / 75.1 / 67.1	65.7 / 60.3 / 75.3 / 67.1	65.7 / 60.4 / 75.3 / 67.1	65.7 / 60.3 / 75.3 / 67.1
Llama-3.1-8B	59.8 / 53.8 / 73.6 / 62.4	59.7 / 53.6 / 73.4 / 62.2	59.7 / 53.7 / 73.4 / 62.3	59.7 / 53.7 / 73.4 / 62.3
Mistral-7B	65.8 / 57.8 / 74.9 / 66.2	65.8 / 57.8 / 74.8 / 66.1	65.8 / 57.8 / 74.8 / 66.1	65.8 / 57.8 / 74.8 / 66.1
Falcon3-7B	60.3 / 55.8 / 70.3 / 62.1	60.3 / 55.9 / 70.4 / 62.2	60.3 / 55.9 / 70.4 / 62.2	60.3 / 56.0 / 70.4 / 62.2
Granite-3.1-8B	64.0 / 51.4 / 75.5 / 63.7	64.0 / 51.7 / 74.8 / 63.5	64.0 / 51.7 / 74.8 / 63.5	64.0 / 51.7 / 74.8 / 63.5

cache preserves the in-context signal needed by these tasks, while the recent-token residual window protects the final local context.

To isolate the role of this residual window, we vary its size W (Figure 2). When the uncompressed window is entirely removed ($W=0$), the underlying full quantization noise is exposed, and distinct performance ordering reappears. As W expands to 128 tokens, the performance of all compressed configurations rapidly converges to the fp16 baseline, remaining practically indistinguishable for larger windows.

4.7 Attention fidelity

Figure 3 shows per-layer cosine similarity between compressed and fp16 key representations. E_8 at 2 bits/dim remains substantially higher than scalar at 3 bits (roughly 0.954 vs. 0.921 average), consistent with the rate-distortion analysis. The shaded range is narrow and smooth across depth, suggesting lattice quantization error is not concentrated in a small subset of layers.

This reconstruction fidelity directly translates to superior preservation of the attention mechanism itself. Figure 3b illustrates the per-layer KL divergence between attention distributions computed with compressed versus fp16 keys. Here, E_8 achieves a remarkably stable and low KL divergence (0.045), effectively halving the error of the 3-bit scalar baseline (0.091). Furthermore, E_8 outperforms KIVI k2v2, which suffers from attention distribution shifts ($KL > 0.200$). This confirms that our lattice geometry preserves the true pre-softmax attention logits more faithfully than existing 2-bit methods, ensuring that improved geometric reconstruction meaningfully carries through to downstream attention behavior.

4.8 Long-context retrieval robustness

To verify that extreme KV cache quantization does not degrade the model’s ability to fetch isolated facts from extended contexts, we conducted a standard single-needle-in-a-haystack retrieval evaluation similar to [Zandieh et al., 2025]. We evaluated retrieval accuracy using a randomly placed fact injected into distractor contexts of varying lengths (4K, 8K, and 16K tokens) across all five models.

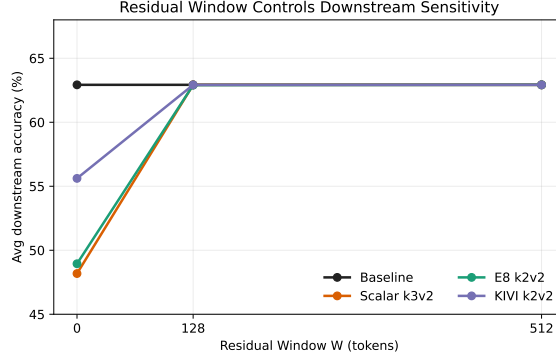


Figure 2: Average downstream accuracy across the core residual-sweep models versus residual window size. At $W = 0$, compression quality differences are visible; by $W = 128$, methods are nearly indistinguishable.

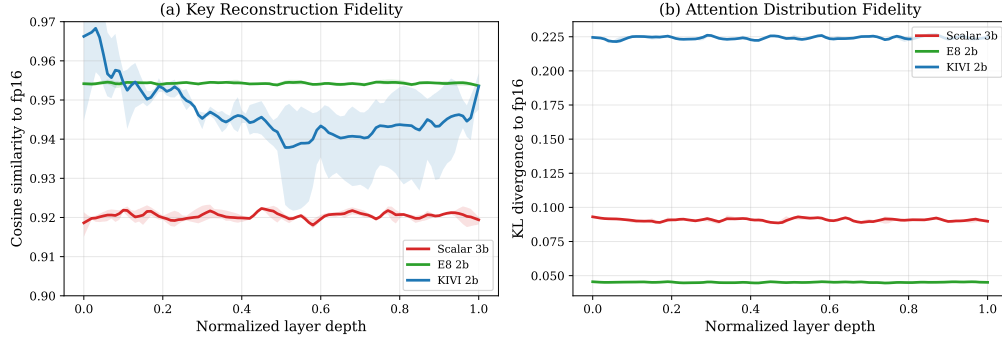


Figure 3: Left: per-layer cosine similarity between compressed and fp16 keys captured from model forward passes. Right: per-layer KL divergence in a controlled attention-distribution probe. Lines show mean over models and shaded bands show the min-max range after normalizing each architecture’s layer index to $[0, 1]$.

E_8 k2v2 achieved 100% retrieval accuracy across all lengths and models, strictly matching the performance of the uncompressed fp16 baseline as well as the competing KIVI and scalar compression methods.

4.9 Ablation studies

Table 5: Compact ablations (mean across core models). (a) E_8 scale calibration. (b) Initial-layer precision sweep.

(a) E_8 scale α									(b) Initial-layer bitwidth					
α	1	4	8	10	12	15	20	30	Cfg	E8-2b	E8-f4-3b	E8-f8-3b	Sc-3b	Sc-f4-4b
MSE	0.0577	0.0045	0.0011	0.0008	0.0007	0.0008	0.0013	0.0024						
cos	0.292	0.727	0.913	0.940	0.953	0.960	0.959	0.952	cos	0.954	0.952	0.950	0.920	0.927

(a) E_8 scale calibration. The scale parameter α controls the dynamic range mapped to the E_8 quantizer. While values that are too small ($\alpha \leq 4$) cause strong clipping and values too large ($\alpha \geq 20$) waste codebook resolution, performance is remarkably robust within the intermediate range. We observe a wide, stable plateau between $\alpha = 8$ and $\alpha = 15$ where MSE and cosine similarity remain stable.

(b) Mixed precision. We also test simple mixed-precision variants that change the first few layers. These variants do not improve the average cosine metric over uniform E_8 k2, so our main configu-

ration keeps a uniform key format across layers. Because the E8 3-bit variant adds a QJL residual correction rather than serving as a pure higher-fidelity E8 codebook, we interpret this ablation only as an implementation-level check, not as evidence that early layers never benefit from additional capacity.

5 Conclusion

We demonstrate that E_8 lattice vector quantization is a practical primitive for low-bit KV-cache key compression. Its fixed geometry-aware codebook enables table-based asymmetric scoring: for each query, dot products with the shared codebook are precomputed once and cached keys are scored directly from indices, avoiding explicit key dequantization in the decode path. Across diverse LLM families and long-context evaluations, this design improves key fidelity and quality per byte while preserving downstream quality and single-sequence decode-throughput parity. At the same time, making compressed-domain scoring efficient beyond this setting, especially for larger batches and prefill-heavy workloads remains an important systems challenge. These results suggest that structured vector codes are a useful building block for KV-compression systems.

References

- Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, M  rouane Debbah,   tienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. The falcon series of open language models, 2023. URL <https://arxiv.org/abs/2311.16867>.
- Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. Quip: 2-bit quantization of large language models with guarantees. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS ’23, Red Hook, NY, USA, 2023. Curran Associates Inc.
- Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, and Beidi Chen. Magicpig: Lsh sampling for efficient llm generation, 2024. URL <https://arxiv.org/abs/2410.16179>.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL <https://arxiv.org/abs/1803.05457>.
- J. H. Conway and N. J. A. Sloane. *Sphere Packings, Lattices and Groups*. Grundlehren der mathematischen Wissenschaften. Springer New York, 3 edition, 1999. doi: 10.1007/978-1-4757-6568-7.
- Jesse Dodge, Maarten Sap, Ana Marasovi  , William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus, 2021. URL <https://arxiv.org/abs/2104.08758>.
- Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. In *Proceedings of the 41st International Conference on Machine Learning*, ICML’24. JMLR.org, 2024.
- Joshua Engels, Eric J. Michaud, Isaac Liao, Wes Gurnee, and Max Tegmark. Not all language model features are one-dimensionally linear, 2025. URL <https://arxiv.org/abs/2405.14860>.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, and Abhishek Kadian et al. The llama 3 herd of models, 2024. URL <https://arxiv.org/abs/2407.21783>.
- Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization. In *Proceedings of the 37th International Conference on Machine Learning*, ICML’20. JMLR.org, 2020.
- Yefei He, Luoming Zhang, Weijia Wu, Jing Liu, Hong Zhou, and Bohan Zhuang. ZipCache: Accurate and efficient KV cache quantization with salient token identification. In *Neural Information Processing Systems*, 2024. doi: 10.52202/079017-2181. URL <https://mlanthology.org/neurips/2024/he2024neurips-zipcache/>.
- Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS ’24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

- Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1):117–128, January 2011. ISSN 0162-8828. doi: 10.1109/TPAMI.2010.57. URL <https://doi.org/10.1109/TPAMI.2010.57>.
- Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, and Florian Bressand et al. Mistral 7b, 2023. URL <https://arxiv.org/abs/2310.06825>.
- 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 Mehdi Rezagholizadeh, Peyman Passban, Soheila Samiee, Vahid Partovi Nia, Yu Cheng, Yue Deng, Qun Liu, and Boxing Chen, editors, *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>.
- Aryan Karmore. Lookat: Lookup-optimized key-attention for memory-efficient transformers, 2026. URL <https://arxiv.org/abs/2601.10155>.
- Haoyang Li, Yiming Li, Anxin Tian, Tianhao Tang, Zhanchao Xu, Xuejia Chen, Nicole Hu, Wei Dong, Qing Li, and Lei Chen. A survey on large language model acceleration based on kv cache management, 2025a. URL <https://arxiv.org/abs/2412.19442>.
- Junyan Li, Yang Zhang, Muhammad Yusuf Hassan, Talha Chafekar, Tianle Cai, Zhile Ren, Pengsheng Guo, Binazir Karimzadeh, Colorado J. Reed, Chong Wang, and Chuang Gan. CommVQ: Commutative vector quantization for KV cache compression. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*. PMLR, 2025b. URL <https://proceedings.mlr.press/v267/li25du.html>.
- 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 *Proceedings of the 38th International Conference on Neural Information Processing Systems, NIPS '24*, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.
- Tengxuan Liu, Shiyao Li, Jiayi Yang, Tianchen Zhao, Feng Zhou, Xiaohui Song, Guohao Dai, Shengen Yan, Huazhong Yang, and Yu Wang. PM-KVQ: Progressive mixed-precision KV cache quantization for long-context LLMs, 2025. URL <https://arxiv.org/abs/2505.18610>.
- Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen (Henry) Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: a tuning-free asymmetric 2bit quantization for kv cache. In *Proceedings of the 41st International Conference on Machine Learning, ICML'24*. JMLR.org, 2024.
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016. URL <https://arxiv.org/abs/1609.07843>.
- Qwen Team, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen2.5 technical report, 2025. URL <https://arxiv.org/abs/2412.15115>.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL <https://arxiv.org/abs/1907.10641>.
- Semyon Savkin, Eitan Porat, Or Ordentlich, and Yuri Polyanskiy. NestQuant: Nested lattice quantization for matrix products and LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 53042–53062. PMLR, 2025. URL <https://proceedings.mlr.press/v267/savkin25a.html>.
- Alina Shutova, Vladimir Malinovskii, Vage Egiastian, Denis Kuznedelev, Denis Mazur, Surkov Nikita, Ivan Ermakov, and Dan Alistarh. Cache me if you must: Adaptive key-value quantization for large language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 55451–55473. PMLR, 13–19 Jul 2025. URL <https://proceedings.mlr.press/v267/shutova25a.html>.
- Prajwal Singhania, Siddharth Singh, Shwai He, Soheil Feizi, and Abhinav Bhatele. Loki: Low-rank keys for efficient sparse attention, 2024. URL <https://arxiv.org/abs/2406.02542>.
- Matt Stallone, Vaibhav Saxena, Leonid Karlinsky, Bridget McGinn, Tim Bula, Mayank Mishra, and Adriana Meza Soria et al. Scaling granite code models to 128k context, 2024. URL <https://arxiv.org/abs/2407.13739>.

- Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: even better llm quantization with hadamard incoherence and lattice codebooks. In *Proceedings of the 41st International Conference on Machine Learning, ICML'24*. JMLR.org, 2024.
- Maryna Viazovska. The sphere packing problem in dimension 8. *Annals of Mathematics*, 185(3):991–1015, 2017. doi: 10.4007/annals.2017.185.3.7. URL <https://annals.math.princeton.edu/2017/185-3/p07>.
- Suqing Wang, Zuchao Li, Luohe Shi, Bo Du, Hai Zhao, Yun Li, and Qianren Wang. From parameters to performance: A data-driven study on llm structure and development, 2025. URL <https://arxiv.org/abs/2509.18136>.
- Xindi Wang, Mahsa Salmani, Parsa Omid, Xiangyu Ren, Mehdi Rezagholizadeh, and Armaghan Eshaghi. Beyond the limits: A survey of techniques to extend the context length in large language models, 2024. URL <https://arxiv.org/abs/2402.02244>.
- Songhao Wu, Ang Lv, Xiao Feng, Yufei Zhang, Xun Zhang, Guojun Yin, Wei Lin, and Rui Yan. Polarquant: Leveraging polar transformation for efficient key cache quantization and decoding acceleration, 2025. URL <https://arxiv.org/abs/2502.00527>.
- Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks, 2024. URL <https://arxiv.org/abs/2309.17453>.
- R. Zamir and M. Feder. On lattice quantization noise. *IEEE Transactions on Information Theory*, 42(4): 1152–1159, 1996. doi: 10.1109/18.508838.
- Amir Zandieh, Majid Daliri, Majid Hadian, and Vahab Mirrokni. Turboquant: Online vector quantization with near-optimal distortion rate, 2025. URL <https://arxiv.org/abs/2504.19874>.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL <https://arxiv.org/abs/1905.07830>.
- Hailin Zhang, Xiaodong Ji, Yilin Chen, Fangcheng Fu, Xupeng Miao, Xiaonan Nie, Weipeng Chen, and Bin Cui. Pqcache: Product quantization-based kvcache for long context llm inference, 2025a. URL <https://arxiv.org/abs/2407.12820>.
- Tianyi Zhang, Jonah Yi, Zhaozhuo Xu, and Anshumali Shrivastava. KV cache is 1 bit per channel: Efficient large language model inference with coupled quantization. In *Proceedings of the 38th International Conference on Neural Information Processing Systems, NIPS '24*, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.
- Yanqi Zhang, Yuwei Hu, Runyuan Zhao, John C. S. Lui, and Haibo Chen. Diffkv: Differentiated memory management for large language models with parallel kv compaction, 2025b. URL <https://arxiv.org/abs/2412.03131>.
- Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS '23*, Red Hook, NY, USA, 2023. Curran Associates Inc.