JoLT: Near-Lossless KV Cache Compression via Joint Rank-Bit Allocation

Anonymous authors
Paper under double-blind review

Abstract

The key-value (KV) cache is the dominant memory bottleneck in long-context language model inference. Existing compression methods apply low-rank factorization or quantization independently, without jointly allocating rank and precision under a shared storage budget. We introduce JoLT, a training-free compressor that treats grouped prefill caches as fourth-order tensors and applies partial Tucker decomposition along the token and feature modes, the two axes that carry low-rank structure, while leaving the head and layer modes intact. A rotated low-bit quantizer captures the truncation residual, and a single Lagrangian dual allocates per-group Tucker ranks and residual bit-widths under a global byte constraint. FlashJoLT replaces the exact token-mode SVD with a randomized approximation that matches JoLT within the free zone at a fraction of the compression cost, and a fused Triton decode kernel evaluates attention directly over the stored factors without materializing dense KV tensors. Across five models from four architecture families, covering multi-head attention, grouped-query attention, and mixture-of-experts architecture, JoLT achieves 2–3$\times$ compression with less than 0.2% perplexity degradation, without retraining. On RULER at 64K context with LLaMA-3.1-8B, retrieval accuracy remains near-lossless through $3\times$ and declines by only 0.90 and 2.40 pp at $4\times$ and $5\times$, respectively. JoLT demonstrates that tensor-aware low-rank decomposition and quantized residuals, unified under a single storage budget, achieve near-lossless KV-cache compression across diverse model architectures without retraining.

1 Introduction

Autoregressive decoding reuses the keys and values of every preceding token at every layer, so inference systems store them in a key-value (KV) cache (Vaswani et al., 2017). The cache grows linearly with batch size, context length, layer count, and the number and dimension of KV heads, and at large batch sizes and long contexts it outgrows the model weights themselves (Pope et al., 2023). Reducing it is the most direct way to serve longer contexts and larger batches on a given device (Kwon et al., 2023).

Existing compression methods shrink the cache through low-rank projection or quantization. Low-rank methods such as xKV (Chang et al., 2026) factor a single two-dimensional unfolding of the cache, capturing correlation across layers but not across other axes. Quantization methods such as KIVI (Liu et al., 2024) and TurboQuant (Zandieh et al., 2026) keep every entry of the full tensor at lower precision and leave its low-rank structure unused. Hybrid methods combine the two but fix the split by hand: GEAR (Kang et al., 2024) corrects a quantized cache with a low-rank term of preset rank, and Palu (Chang et al., 2025) and STAR-KV (Bhatnagar et al., 2026) quantize their low-rank latents at a preset bit-width after the ranks are chosen. None of them decides how much of a fixed byte budget to spend on rank and how much on precision.

Krishnan & Schulz (2026) treat grouped layer caches as fourth-order tensors over layers, heads, tokens, and features, and measure the singular-value spectrum of each mode. The head and layer modes are index-like, with nearly flat spectra (a dynamic range of 1.4 on the head mode of Mistral-7B), so truncating them incurs a large error that no budget can recover. The token and feature modes carry the low-rank structure, and values are about $2.4\times$ harder to compress than keys because their feature spectrum is flat. These findings determine both where to truncate and how to split the budget.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

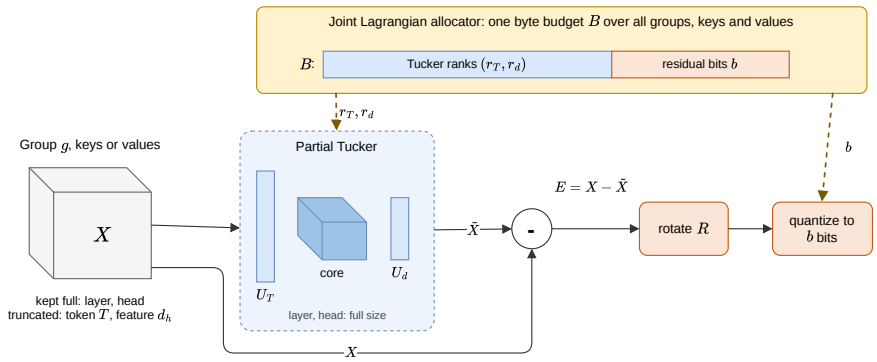

Figure 1: Overview of JoLT. Each cell (one layer group's keys or values) is compressed in two stages, with a single Lagrangian allocator splitting the byte budget B across Tucker ranks and residual bit-widths for all cells jointly.

We introduce **JoLT** (Joint Lagrangian Tucker), a training-free compressor built on this design (Figure 1). JoLT groups consecutive layers whose token subspaces align, so that one pair of token and feature bases serves the whole group, and applies a partial Tucker decomposition that truncates only the token and feature modes. The truncation discards a spectral tail that is large especially for values, so JoLT stores it as a low-bit residual after a random orthogonal rotation spreads channel outliers across coordinates. A single Lagrangian dual then allocates Tucker ranks and residual bit-widths to every group, and to keys and values separately, under one global byte budget. Each byte therefore goes to the rank or the bit that removes the most reconstruction error.

Two further components make JoLT practical. FlashJoLT replaces the exact token-mode SVD, which dominates compression time, with a randomized SVD (Halko et al., 2011) that computes only the leading singular values. It estimates the energy of the remaining ones from the tensor norm so that the allocator's rank-bit decisions stay correct. Decompressing the cache to dense tensors at decode time would erase the memory savings, so a fused Triton (Tillet et al., 2019) kernel evaluates attention directly over the stored factors and packed residuals.

Across five models from four families, spanning multi-head (Vaswani et al., 2017) and grouped-query attention (Ainslie et al., 2023) as well as a mixture-of-experts architecture (Muennighoff et al., 2025), JoLT compresses the cache 2–3$\times$ with less than 0.2% perplexity drift and no retraining. We call this band the free zone. The free zone carries over to downstream tasks: on RULER (Hsieh et al., 2024) at 64K context with LLaMA-3.1-8B, retrieval stays near-lossless through $3\times$ and drops by only 0.90 and 2.40 pp at $4\times$ and $5\times$ (Sections 4 and 4.3).

Our contributions are:

1. **Joint rank-precision allocation.** A compression formulation that combines partial Tucker truncation of the token and feature modes with a rotated low-bit residual, and allocates ranks and bit-widths jointly under one byte budget (Section 3).
2. **FlashJoLT.** A randomized token-mode decomposition with tail-mass accounting method that keeps the allocation correct and matches JoLT in the free zone at a fraction of the compression cost (Section 3.1).
3. **A cross-architecture free zone.** Evidence on five models that 2–3$\times$ compression is near-lossless without retraining and holds on downstream reasoning and long-context retrieval (Sections 4 and 4.3).
4. **Fused decode kernel.** A fused Triton kernel evaluates attention over the compressed representation without dense reconstruction, reducing peak KV decode memory by 1.43 to $4.73\times$ across 2–8$\times$ targets (Section 4.5).

2 Related Work

KV-cache quantization. Quantization compresses the cache by lowering the bit-width of every stored entry. KIVI applies tuning-free asymmetric 2-bit quantization with per-channel keys and per-token values (Liu et al., 2024). TurboQuant composes a random rotation with an MSE-optimal scalar quantizer and a 1-bit residual projection to approach the rate-distortion bound (Zandieh et al., 2026). KVQuant quantizes keys before the rotary embedding and handles per-channel outliers (Hooper et al., 2024), sharing JoLT's pre-RoPE motivation but fixing the bit-width in advance with no rank component. Each quantizer operates at one fixed ratio set by its bit-width, with no way to target intermediate values. JoLT covers the 2–$10\times$ range continuously by jointly optimizing rank and precision at any target byte budget. The two approaches are complementary: the model weights and the Tucker factors themselves can be quantized for additional compression beyond what either method achieves alone.

Low-rank and structural KV compression. Palu factorizes per-head-group projection weights offline and caches the low-rank latents, optionally quantized after a fused Hadamard rotation (Chang et al., 2025). xKV projects grouped cross-layer feature blocks onto a shared singular-vector basis (Chang et al., 2026). KQ-SVD decomposes the attention matrix $QK^\top$ rather than the cache itself (Lesens et al., 2026). ReCalKV combines head-reordered SVD for keys with calibrated value projections (Yan et al., 2025). STAR-KV picks per-head key ranks by learned soft thresholding and quantizes the latents at a fixed mixed-precision split (Bhatnagar et al., 2026). All of these are matrix methods: each factors a single two-dimensional slice of the cache and never sees the full multi-linear structure across heads, tokens, features, and layers. Palu and STAR-KV add quantization to their latents, but both fix the bit-width by hand after the ranks are set. GEAR (Kang et al., 2024) takes the opposite approach, pairing a quantized base with a low-rank correction and a sparse outlier term, but fixes its rank, sparsity ratio, and bit-width uniformly across layers. In all cases, rank and precision are set independently rather than traded off under a shared budget. Token-eviction methods are orthogonal in mechanism and can in principle be combined with JoLT (Zhang et al., 2023; Xiao et al., 2024; Li et al., 2024).

Tensor decompositions and rate allocation. JoLT computes its partial Tucker decomposition via the sequentially truncated higher-order SVD (ST-HOSVD) (Vannieuwenhoven et al., 2012), a sequential variant of the HOSVD (De Lathauwer et al., 2000) applied to the Tucker format (Tucker, 1966). CP (Kolda & Bader, 2009), tensor train (Oseledets, 2011), and t-SVD (Kilmer & Martin, 2011) are alternative tensor formats. JoLT uses Tucker because it can leave the index-like head and layer modes untouched while truncating only the compressible ones. The rotated low-bit residual draws on incoherence processing (Chee et al., 2023), the principle behind rotation-based weight quantization (Ashkboos et al., 2024; Liu et al., 2025), applied here to the truncation residual of a tensor factorization rather than to weights. The Lagrangian allocation that ties the two stages together is a standard relaxation of a constrained resource-allocation problem, but has not previously been applied jointly to per-mode Tucker ranks and residual bit-widths for KV-cache compression.

3 Method: Joint Lagrangian Tucker (JoLT)

The spectral structure described in Section 1 determines JoLT's design: truncate only the token and feature modes, leave the head and layer modes at full rank, and give keys and values separate budgets so that the harder value tensor draws more bytes. JoLT compresses the KV cache produced during prefill in two stages, a partial Tucker decomposition and a rotated low-bit residual, and allocates storage to each stage with a single Lagrangian optimizer (Figure 1).

We partition the L layers into G contiguous groups by clustering layers whose token-mode singular vectors are well-aligned, so that a single set of factor bases can serve the group without cross-layer interference. For each layer, we compute the top eight token-mode singular vectors of its key cache and score consecutive layers by the mean squared cosine between their bases. A new group starts whenever the score falls below the median score across the network.[1] The grouping is computed

[1]The size bounds take priority: a new group opens only once the current one has at least two layers, a group is closed at eight layers regardless of the score, and a trailing group of one layer is merged into its predecessor, so the last group can reach nine layers.

from the cache being compressed, with no separate calibration data. On Mistral-7B-v0.3 at T=1024, the 32 layers form 11 groups.

Keys are decomposed before the rotary position embedding (Su et al., 2024) because post-RoPE keys show 48 to 62% higher reconstruction error at matched storage (Krishnan & Schulz, 2026). The rotation is applied to the reconstructed keys at decode time.

Within each group g, we treat keys and values as separate compression targets $t \in \{\mathrm{K}, \mathrm{V}\}$. Each group's key or value cache forms a fourth-order tensor $\mathcal{X}_g \in \mathbb{R}^{|g| \times n_h \times T \times d_h}$, where $|g|$ is the number of layers in the group, n_h is the number of KV heads, T is the number of cached prefill tokens, and d_h is the per-head feature dimension. The partial Tucker decomposition truncates the token and feature modes to ranks r_T and r_d while keeping the layer and head modes at full rank with identity factors. We call each pair (g, t) a *cell* and compress every cell with the three stages below.

Partial Tucker backbone. The partial Tucker approximation $\tilde{X}(r_T, r_d)$ computes the mode-k singular vectors of the token and feature unfoldings via the sequentially truncated higher-order SVD (ST-HOSVD) (Vannieuwenhoven et al., 2012) and truncates them to ranks r_T and r_d, while keeping identity factors on the layer and head modes. The token mode is truncated first, and the feature-mode factor is then computed on the projected tensor, so the feature truncation sees only the energy that survived the token step. The resulting core has shape $|g| \times n_h \times r_T \times r_d$, and two factor matrices $U_T \in \mathbb{R}^{T \times r_T}$ and $U_d \in \mathbb{R}^{d_h \times r_d}$ store the token-mode and feature-mode bases shared across every head and every layer in the group. With identity factors on the layer and head modes, a single ST-HOSVD pass directly constructs the partial Tucker approximation without iterative refinement.

The ST-HOSVD approximation is quasi-optimal in Frobenius norm: its error is at most $\sqrt{2}$ times the minimum over all approximations with token rank at most r_T and feature rank at most r_d. This follows from the ST-HOSVD error bound (Vannieuwenhoven et al., 2012, Thm. 6.5) and the Eckart–Young theorem on each of the two truncated unfoldings. Riemannian optimization on the fixed-rank Tucker manifold (Heidel & Schulz, 2018) could tighten this bound, but we keep the single pass for speed and let the residual stage capture the full truncation error.

Rotated low-bit residual. The truncation discards a spectral tail that is large for values, so moderate Tucker ranks alone cannot reach near-lossless fidelity. Rather than spending additional rank on diminishing returns, JoLT stores the truncation residual $E = \mathcal{X}_g - \tilde{X}(r_T, r_d)$ at low bit-width.

We apply a random orthogonal rotation along the feature axis to spread residual energy evenly across channels, so that no single outlier channel dominates the quantizer's dynamic range. The rotation matrix is the $d_h \times d_h$ factor from the QR decomposition of a seeded Gaussian, shared by every cell and regenerated from its seed at decode, so it costs no storage. We then quantize the rotated entries uniformly at $b \in \{0, 2, 4, 8\}$ bits per element with one fp32 scale per row, where b=0 means no residual is stored. Decoding inverts the rotation and adds the dequantized residual to the low-rank reconstruction.

Joint Lagrangian allocation. The backbone and the residual compete for the same bytes, and the right split depends on the spectrum of each cell. A fixed policy that always applies four residual bits, or never stores a residual, cannot adapt to this per-cell variation. JoLT therefore allocates ranks and residual bits jointly under a single byte budget.

Consider a single cell (g, t) and let c denote the bytes per stored scalar (c=2 for bf16). A choice of token rank r_T, feature rank r_d, and residual bit-width b costs

$$s_{g,t}(r_T, r_d, b) \;=\; \underbrace{\big(|g|\, n_h\, r_T r_d + T r_T + d_h r_d\big)\, c}_{\text{Tucker core and factors}} \;+\; \underbrace{[\,b > 0\,]\, \frac{b + 32/d_h}{8}\, |g|\, n_h\, T\, d_h}_{\text{packed rotated residual}} \qquad (1)$$

bytes. The two identity factors on the layer and head modes are negligible and omitted. The residual term accounts for one fp32 scale per residual row, giving an effective bit-width of $b + 32/d_h$. At d_h=128 and b=4 this is 4.25 bits per element, and at b=0 no residual and no scales are stored, so the term vanishes.

The reconstruction error is modeled as

$$e_{g,t}(r_T, r_d, b) \;\approx\; \varepsilon^2(b) \;\cdot\; \tau_{g,t}(r_T, r_d). \qquad (2)$$

Here $\tau_{g,t}(r_T, r_d)$ is the relative Frobenius mass that the partial Tucker truncation discards at ranks (r_T, r_d), computed as the sum over the two truncated modes of the discarded mode-k singular-value mass divided by $\|\mathcal{X}_g\|_F^2$. The factor $\varepsilon^2(b)$ is the fraction of that discarded mass the rotated b-bit quantizer fails to recover, calibrated once on a Gaussian round-trip with $\varepsilon^2(0)=1$. At $d_h=128$ the calibration gives $\varepsilon^2(2) \approx 0.59$, $\varepsilon^2(4) \approx 0.014$, and $\varepsilon^2(8) \approx 4 \times 10^{-5}$. This multiplicative factorization is the key modeling choice: rank reduces the truncation tail τ, and bits reduce the residual factor $\varepsilon^2(b)$, making the two stages commensurable on a single error scale.

Solving the allocation. Given a total byte budget B for the compressed prefill cache, JoLT minimizes the summed error subject to the summed cost:

$$\min_{\{(r_T, r_d, b)_{g,t}\}} \sum_{g,t} e_{g,t}(r_T, r_d, b) \quad \text{s.t.} \quad \sum_{g,t} s_{g,t}(r_T, r_d, b) \le B. \tag{3}$$

We solve this through its Lagrangian relaxation,

$$\mathcal{L}(\lambda) \;=\; \sum_{g,t} \Big[\, e_{g,t}(r_T, r_d, b) + \lambda\, s_{g,t}(r_T, r_d, b) \,\Big], \tag{4}$$

which separates across cells. Because r_T, r_d, and b lie on finite grids, Equation (3) is a discrete separable allocation problem and requires no differentiability.

For fixed λ, each cell independently minimizes $e + \lambda s$. We enumerate $b \in \{0, 2, 4, 8\}$ and, for each b, greedily increase ranks starting from $r_T = r_d = 1$. At each step the solver compares the marginal error reduction per byte for incrementing r_T by one against incrementing r_d by one, and takes whichever gives the steeper descent. A mode's rank increases by one whenever $\varepsilon^2(b)\, \sigma_{k,r_k+1}^2 / \|\mathcal{X}_g\|_F^2 > \lambda\, \Delta s$, where $\sigma_{k,\cdot}$ are the singular values of the unprojected mode-k unfolding and Δs is the byte increment from Equation (1). The search stops when a full sweep over both modes makes no change, and the cell keeps the b with the smallest $e + \lambda s$. This rule is exact for a single truncated mode but is a coordinate-wise heuristic with two modes because the core cost contains the bilinear term $|g|\, n_h\, r_T r_d$. With exact per-cell minimization, the multiplier theorem of Everett (1963) guarantees optimality for the budget actually attained, following the standard approach in discrete bit allocation (Shoham & Gersho, 1988). Since total cost is non-increasing in λ, we find the multiplier by geometric bisection, stopping within 0.5% of the target budget or after 50 iterations. The multiplier λ acts as a global shadow price on memory, jointly balancing rank and residual precision across all cells.

Surrogate limitations. The surrogate of Equation (2) charges every unit of discarded Frobenius mass equally, whether it remains as structured truncation error or as quantization noise, and it does not model the asymmetric effect of key errors through the softmax. In the free zone this approximation is accurate, but at higher compression on LLaMA-2 it makes extra key rank appear cheaper than key residual bits of the same byte cost. Section 3.1 introduces a token-rank cap that guards against this failure mode, and Section 4.4 quantifies the effect.

3.1 Fast variant: FlashJoLT

The exact token-mode SVD dominates compression time. The token-mode unfolding of $\mathcal{X}_g \in \mathbb{R}^{|g| \times n_h \times T \times d_h}$ has shape $T \times (|g|\, n_h\, d_h)$, and its economy SVD scales as $O\big(T\,(|g|\, n_h\, d_h)^2\big)$ when T exceeds $|g|\, n_h\, d_h$, which is the typical regime at long contexts. FlashJoLT replaces this SVD with a randomized approximation (Halko et al., 2011) that computes only the leading q_{cap} token-mode singular vectors. The feature-mode SVD is kept exact because d_h is small (128), and the rest of the pipeline, including the Lagrangian allocator, is unchanged.

The allocator requires the full spectrum to price rank against residual bits. Since the randomized SVD computes only the top q_{cap} singular values, FlashJoLT estimates the remaining energy as $\|\mathcal{X}_g\|_F^2$ minus the energy of the computed singular values and appends it as a single synthetic tail entry. The allocator therefore sees the correct total energy and makes valid rank-bit decisions on the truncated spectrum.

The cap q_{cap} controls how many singular vectors are computed. With C the target compression ratio, the policy is

$$q_{\text{cap}} \;=\; \min\big(\max(q_{\min}(C),\, \lceil T/32 \rceil),\, 512\big), \qquad q_{\min}(C) = \begin{cases} 32 & C \le 4 \\ 64 & C > 4, \end{cases} \tag{5}$$

so the cap starts at $q_{\min}(C)$ for short contexts, grows linearly with T, and saturates at 512. The cap also serves as a guard against the surrogate limitation described in Section 3: by bounding the token rank, it prevents the allocator from over-spending on key rank at the expense of key residual bits, which matters on LLaMA-2 past the free zone (Appendix A). Calibration details and the LLaMA-2 analysis are in Appendix A.

Quality is essentially unchanged in the free zone. On Mistral at $T{=}1024$, the reconstruction-error gap between FlashJoLT and JoLT is $|\Delta_K| \leq 0.0093$ and $|\Delta_V| \leq 0.0011$ across 2–3× (Appendix A). Timing both compressors over the full compression pipeline, including layer grouping, gives end-to-end speedups of $1.04\times$ to $3.97\times$ depending on cache length and target ratio (Appendix A).

4 Experiments

4.1 Setup

We evaluate JoLT on five models spanning four architecture families: Mistral-7B-v0.3 (Jiang et al., 2023) (GQA, $n_{\text{kv}}{=}8$, 32 layers), LLaMA-2-13B (Touvron et al., 2023) (MHA, $n_{\text{kv}}{=}40$, 40 layers), LLaMA-3.1-8B (Grattafiori et al., 2024) (GQA, $n_{\text{kv}}{=}8$, 32 layers), Qwen2.5-14B (Qwen Team, 2024) (GQA, $n_{\text{kv}}{=}8$, 48 layers), and OLMoE-1B-7B-0924 (Muennighoff et al., 2025) (MoE with MHA, $n_{\text{kv}}{=}16$, 16 layers). Mistral and LLaMA-2 are the primary evaluation pair and appear in every experiment.

All models run in bf16 on A100 GPUs (see the Reproducibility Statement) with float32 decomposition numerics. At context length n, the prompt is split into two halves: the first $n/2$ tokens serve as the prefill cache that JoLT compresses ($T = n/2$), and the second half is scored against the compressed cache to produce the perplexity. Perplexity is computed on a deterministic WikiText-2 (Merity et al., 2017) and C4 (Raffel et al., 2020) mixture and reported as the mean over seeds $\{0, 1, 2\}$ unless otherwise noted. Across-seed standard deviation is at most 0.009 PPL on every free-zone cell.

Unless stated otherwise, results use JoLT (exact SVD). FlashJoLT is used for LLaMA-3.1 and Qwen in the perplexity tables and for all RULER experiments.

4.2 Perplexity: the near-lossless free zone

We call the 2–3× compression band the *near-lossless free zone*: the regime where perplexity drift stays below 0.2% on every model tested.

Table 1 reports perplexity at $n{=}1024$ ($T{=}512$) on all five models. Every free-zone cell stays below the 0.2% threshold, with the largest drift at $+0.17\%$ (Qwen at $3\times$). The free zone holds at context lengths 512 and 2048 as well (Appendix B). Beyond the free zone the models diverge: Mistral and OLMoE degrade gradually at $4\times$ ($+5.59\%$ and $+3.00\%$), while LLaMA-2 jumps sharply to $+27.28\%$ (Figure 2a). The degradation pattern is model-specific, not attention-type-specific, and a detailed breakdown appears in Appendix B.

Comparison with baselines. Table 2 compares JoLT against six low-rank baselines, three fixed-rate quantizers, and two 8-bit baselines on Mistral at $n{=}1024$, evaluated on a separate 150-chunk grid (LLaMA-2 results in Appendix E, Table 16). At $2\times$, JoLT drifts $+0.01\%$ while the best low-rank baseline (KQ-SVD) drifts $+5.27\%$. Fixed-rate 8-bit quantization (int8 per-channel and fp8 e4m3) is also near-lossless at $2\times$, but has no operating point above $2\times$. Near $4\times$, the dedicated quantizers at their native ratios outperform JoLT: TurboQuant at $+0.28\%$ and KIVI at $+1.70\%$ against JoLT's $+5.65\%$. JoLT's advantage over every evaluated baseline therefore lies in the band above $2\times$ and below roughly $4\times$, which is exactly the free zone (Figure 2b).

4.3 Downstream evaluation

Long-context retrieval (RULER, 64K). On a 4-task subset of RULER (Hsieh et al., 2024) (50 samples per task, recall-based accuracy with partial credit), FlashJoLT is near-lossless through $3\times$ at 64K context on LLaMA-3.1-8B (Figure 3). At $2\times$ the compressed model scores 91.20 against the 90.97 uncompressed baseline ($\Delta = +0.23$ pp), within noise. At $3\times$ the delta is -0.10 pp, also

Table 1: Perplexity at context length n=1024 (T=512) on five models spanning four families. $\Delta\%$ is relative to each model's own uncompressed baseline. All cells are 3-seed means except where noted.

Model	baseline	2×	3×	4×	5×
Mistral-7B (GQA)	6.28	6.28 (−0.02%)	6.29 (+0.04%)	6.64 (+5.59%)	7.01 (+11.56%)
LLaMA-2-13B (MHA)	5.39	5.39 (−0.02%)	5.39 (−0.01%)	6.86 (+27.28%)	9.07 (+68.29%)
OLMoE-1B-7B† (MoE)	7.26	7.27 (+0.02%)	7.27 (+0.11%)	7.48 (+3.00%)	–
LLaMA-3.1-8B‡ (GQA)	7.28	7.28 (−0.01%)	7.29 (+0.12%)	7.96 (+9.33%)	11.02 (+51.27%)
Qwen2.5-14B* (GQA)	6.65	6.65 (+0.02%)	6.66 (+0.17%)	7.96 (+19.63%)	9.55 (+43.52%)

† WikiText-2 only, 100 chunks, ratios 2–4 only. ‡ FlashJoLT. * FlashJoLT, single seed (seed 0), 100 chunks.

Table 2: Cross-method perplexity on Mistral-7B at n=1024 (150 chunks, 3-seed mean). $\Delta\%$ is relative to the grid's uncompressed baseline of 6.76. Quantizers operate at a single fixed ratio. This grid uses a different evaluation subset from Table 1. LLaMA-2 results in Appendix E.

Low-rank methods

Method	2×	3×	4×
JoLT	6.76 (+0.01)	6.76 (+0.04)	7.14 (+5.65)
FlashJoLT	6.76 (+0.00)	6.79 (+0.38)	7.14 (+5.59)
xKV	7.22 (+6.75)	7.65 (+13.2)	8.07 (+19.4)
Palu	7.24 (+7.06)	7.61 (+12.6)	7.85 (+16.1)
ReCalKV	7.25 (+7.27)	7.65 (+13.1)	7.90 (+16.9)
KQ-SVD	7.12 (+5.27)	7.35 (+8.67)	7.60 (+12.4)

Quantizers

Method	ratio	PPL ($\Delta\%$)
KIVI 2-bit	4.20×	6.88 (+1.70)
TurboQuant	3.82×	6.78 (+0.28)
int4 p.c.	3.97×	6.93 (+2.42)
int8 p.c.	1.99×	6.76 (−0.00)
fp8 e4m3	2.00×	6.76 (+0.01)

within noise. Beyond the free zone, retrieval degrades smoothly: 4× costs −0.90 pp and 5× reaches 88.57 ($\Delta = -2.40$ pp), with the full ladder monotone.[2]

At 5×, the RULER score drops by only 2.40 pp despite a +51.27% perplexity degradation at the same compression ratio (Table 1). Perplexity scores every next token and is dominated by the hardest positions, while RULER scores a small number of retrieval-critical tokens. The two metrics therefore measure different slices of model quality and need not degrade together. Per-context-length results for Mistral and additional LLaMA-3.1 arms appear in Appendix C.

Reasoning (GSM8K). On GSM8K (Cobbe et al., 2021) with 8-shot chain-of-thought prompting (Wei et al., 2022) and exact-match scoring (n=1319 per cell), JoLT is near-lossless at 2× and 3× on both Mistral and LLaMA-2 (Table 7 in Appendix C). All free-zone deltas fall within their 95% confidence intervals of zero, with the largest shift at −1.21 pp (Mistral 3×). At 4× accuracy drops sharply on Mistral (−8.49 pp), consistent with the perplexity cliff, while LLaMA-2 degrades more gently (−2.20 pp). A matched-memory comparison against int4, KIVI, and TurboQuant appears in Appendix C.

Real-world long context (LongBench, 16K). On six LongBench (Bai et al., 2024) tasks at 16K context on Mistral-7B (multi-document QA, summarization, and few-shot learning, 200 examples per task, seed 0), FlashJoLT stays within 0.52 pp of the uncompressed pooled score through 4×. The pooled mean drifts by −0.09, +0.05, and −0.52 pp at 2×, 3×, and 4×, with no single task drifting by more than 0.91 pp at any ratio (Appendix C, Table 10). The evidence covers one model and one seed.

4.4 Ablations

We ablate the design choices on Mistral-7B-v0.3. Full grids across $n \in \{512, 1024, 2048\}$ are in Appendix D (Tables 12 and 13). Perplexities below are at n=1024 on the ablation evaluation subset unless stated.

[2] 95% CIs from a seed-stratified percentile bootstrap (N_{boot}=200,000 resamples × 5 independent RNG streams). All RULER arms use 3 dataset seeds and 600 rows per arm.

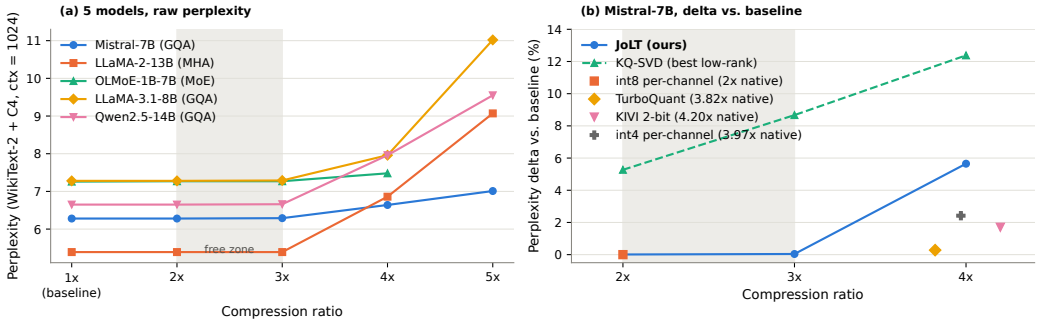

Figure 2: (a) Perplexity versus compression ratio at n=1024 on all five models (Table 1). The 2–3× free zone is shaded. (b) Perplexity change on Mistral-7B for JoLT and baselines on the 150-chunk cross-method grid (Table 2). Quantizers are shown at their native ratio.

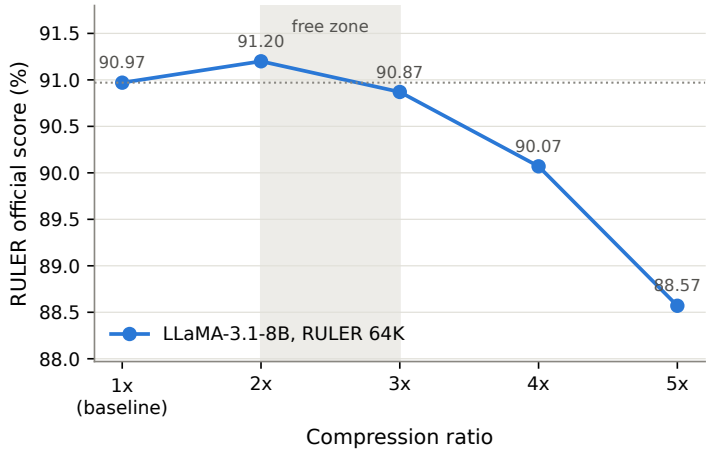

Figure 3: RULER score versus compression ratio on LLaMA-3.1-8B at 64K context (FlashJoLT, 3 dataset seeds, 600 rows per arm). The 2–3× free zone is shaded.

Rotated residual. Dropping the residual entirely (rank-only compression) raises 2× perplexity from 6.54 to 7.27, a gap that additional rank cannot close.

Allocation strategy. The joint Lagrangian allocation matches K/V-decoupled allocation at 2× and trails it at 4× (6.93 against 6.72), where the budget is generous enough that rebalancing between keys and values has little effect. At 8× the joint solve pulls ahead (8.14 against 8.42 at n=1024, 0.45 PPL gap at n=512), where the tighter budget makes K/V rebalancing valuable. At n=2048 the two stay within 0.17 PPL of each other.

Per-group ranks. Per-group rank allocation beats uniform rank at all 15 grid cells (ratios 2, 3, 4, 6, 8 at three context lengths), with a peak advantage of 0.97 PPL at n=512, 4×.

Residual bit-width. Residual bits saturate at four: the step from 4 to 8 bits improves perplexity by less than 0.001 at 2×.

A further ablation on FlashJoLT versus JoLT parity appears in Appendix A.

4.5 FUSED DECODE KERNEL

Reconstructing dense keys and values from the compressed factors at each decode step would erase the memory savings. The FlashJoLT decode kernel evaluates attention directly over the stored factors and packed residual, so no dense KV buffer is ever written to global memory.

The kernel follows the flash-decoding pattern: each program owns one KV head and a range of token blocks and carries one online-softmax state across them. For each block, it rebuilds the key tile on chip from the token factor, core, and feature factor. The residual is unpacked from its low-bit codes and derotated in the same pass, and RoPE is applied to the reconstructed tile before scoring against the query. Values are handled differently: softmax weights are accumulated against the value token factor in rank space, only the value residual is decoded per block, and the rank-sized partial state is expanded through the core and feature factor once per head after the block states are merged. Newly generated tokens live in a small dense tail that shares the same softmax state.

On the key side, the position-dependent RoPE does not commute with the token-mode factorization, so keys must be fully reconstructed per tile rather than scored in rank space as values are. This is the main source of latency overhead and is discussed further in Section 5.

Memory. On Mistral-7B-v0.3 at context length $8{,}192$, peak decode memory of the KV representation is $1.43\times$, $2.40\times$, $3.61\times$, and $4.73\times$ below the dense bf16 cache at the $2\times$, $3\times$, $5\times$, and $8\times$ compression targets.[3]

Latency. The fused kernel is slower than uncompressed attention at every measured point. In a microbenchmark of one Mistral-7B attention layer at batch 1 on an A100 80GB, its latency is $7.5\times$ to $65.9\times$ the dense baseline (PyTorch scaled-dot-product attention over a resident bf16 cache) across contexts 512 to $8{,}192$ at $2\times$, $3\times$, and $5\times$ targets. Against a reconstruct-then-attend path, the fused kernel's latency ratio is $0.7\times$ to $2.2\times$, falling below 1 only at context $8{,}192$. End-to-end task latency confirms the overhead: on RULER NIAH with Mistral-7B at $8{,}192$ context (median over 250 tasks, including compression and per-step reconstruction), the compressed path takes 20.7s and 18.2s at $2\times$ and $3\times$ against 1.6s uncompressed. At batch 1 the resident dense cache is read faster than the factors can be contracted, so the kernel's current value is memory capacity, not speed. We did not measure tokens-per-second throughput or batched serving, where the memory savings would translate into higher concurrency.

5 CONCLUSION

JoLT shows that low-rank compression is competitive with quantization in the near-lossless regime. Across five models, partial Tucker decomposition paired with a rotated low-bit residual holds a 2–$3\times$ free zone with less than 0.2% perplexity drift, and a single Lagrangian dual makes this work at any target byte budget without per-model tuning. The two approaches are complementary: quantizing the Tucker factors or the model weights can push compression further.

The fused decode kernel delivers the memory savings at serving time, reducing peak KV memory by up to $4.73\times$, but the latency cost is substantial. The main bottleneck is RoPE: because the position-dependent rotation does not commute with the token-mode factorization, keys must be fully reconstructed per tile rather than scored in rank space as values are. Folding the rotation into the factor scoring would remove this per-tile reconstruction and is the primary open problem on the kernel side.

Two further limitations point to future work. The Lagrangian surrogate prices all Frobenius mass equally and does not model how key errors pass through the softmax, which leads to over-allocation of key rank on some models past the free zone. A tighter surrogate or an improved cap policy would extend the near-lossless range beyond $3\times$. The current pipeline compresses the prefill cache in one global pass, which suits prompt caching but does not extend to streaming autoregressive generation, where efficient online rank updates without full SVD recomputation remain an open problem.

[3] The $8\times$ target overshoots to $9.8\times$ achieved ratio.

REPRODUCIBILITY STATEMENT

Source code for JoLT, FlashJoLT, and the baseline harnesses used in this paper is provided as supplementary material. The method is fully specified in Section 3 (layer grouping, the partial Tucker backbone, the rotated residual, and the allocator in Equations (1) to (4)), and the randomized variant and its cap policy are specified in Section 3.1 and Appendix A. Evaluation protocols, seeds, and evaluated-token counts are stated alongside each experiment (Sections 4 and 4.3; Appendices B, C, and E); all five models are public checkpoints and all datasets (WikiText-2, C4, GSM8K, RULER, LongBench) are public. GPU experiments ran on NVIDIA A100 GPUs (40 GB PCIe, 40 GB SXM4, and 80 GB PCIe) and, for the 4K and 8K LLaMA-3.1-8B RULER rows of Table 9, on one RTX 3090, under PyTorch 2.5.1 with CUDA 12.1, with models in bf16 and decomposition numerics in float32; the fused-kernel memory measurements reproduce bit-identically across the two A100 PCIe SKUs.

AI USE STATEMENT

The authors used generative AI tools during this work. LLM-based coding assistants were used to assist with software implementation and experiment scaffolding, generative AI tools were used to assist with manuscript drafting and editing, and AI-assisted search was used to retrieve related work and to check citation records against publisher metadata. Generative AI was not used to formulate the research questions, design the proposed method or experimental protocol, analyze or interpret experimental results, or formulate the paper's scientific claims. All AI-assisted code and text were reviewed and verified by the authors. The authors take full responsibility for the final content of the paper.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4895–4901, 2023.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-free 4-bit inference in rotated LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

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*, 2024.

Priyansh Bhatnagar, Ashkan Moradifirouzabadi, Se-Hyun Yang, SeungJae Lee, Jungwook Choi, and Mingu Kang. STAR-KV: Low-rank KV cache compression via soft thresholding for adaptive rank control. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. arXiv:2606.08382.

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 *International Conference on Learning Representations (ICLR)*, 2025.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Hung-Yueh Chiang, Yash Akhauri, Xilai Dai, Huiqiang Jiang, Yucheng Li, Luis Ceze, Kai-Chiang Wu, and Mohamed S. Abdelfattah. xKV: Cross-layer KV-cache compression via aligned singular vector extraction. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. QuIP: 2-bit quantization of large language models with guarantees. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Karl Cobbe et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

Hugh Everett. Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, 11(3):399–417, 1963.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53 (2):217–288, 2011.

Gennadij Heidel and Volker Schulz. A Riemannian trust-region method for low-rank tensor completion. *Numerical Linear Algebra with Applications*, 25(6):e2175, 2018.

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 *Advances in Neural Information Processing Systems*, volume 37, 2024.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *Conference on Language Modeling (COLM)*, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

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, pp. 305–321. PMLR, 2024.

Misha E. Kilmer and Carla D. Martin. Factorization strategies for third-order tensors. *Linear Algebra and its Applications*, 435(3):641–658, 2011.

Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3): 455–500, 2009.

Rahul Krishnan and Volker Schulz. Tensor decomposition of transformer key-value caches: Spectral structure and format comparison. arXiv preprint arXiv:2609.28029, 2026.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, pp. 611–626, 2023. doi: 10.1145/3600006.3613165.

Damien Lesens, Beheshteh T. Rakhshan, and Guillaume Rabusseau. KQ-SVD: Compressing the KV cache with provable guarantees on attention fidelity. In *Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2026.

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 *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. SpinQuant: LLM quantization with learned rotations. In *International Conference on Learning Representations (ICLR)*, 2025.

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 *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations (ICLR)*, 2017.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. OLMoE: Open mixture-of-experts language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Ivan V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5): 2295–2317, 2011.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. In *Proceedings of Machine Learning and Systems (MLSys)*, 2023.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text Transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Yair Shoham and Allen Gersho. Efficient bit allocation for an arbitrary set of quantizers. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(9):1445–1453, 1988.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Philippe Tillet, H. T. Kung, and David Cox. Triton: An intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages (MAPL)*, pp. 10–19, 2019.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Ledyard R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3): 279–311, 1966.

Nick Vannieuwenhoven, Raf Vandebril, and Karl Meerbergen. A new truncation strategy for the higher-order singular value decomposition. *SIAM Journal on Scientific Computing*, 34(2):A1027–A1052, 2012.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

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 (NeurIPS)*, 2022.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations (ICLR)*, 2024.

Xianglong Yan, Zhiteng Li, Tianao Zhang, Haotong Qin, Linghe Kong, Yulun Zhang, and Xiaokang Yang. ReCalKV: Low-rank KV cache compression via head reordering and offline calibration. *arXiv preprint arXiv:2505.24357*, 2025.

Amir Zandieh, Majid Daliri, Majid Hadian, and Vahab Mirrokni. TurboQuant: Online vector quantization with near-optimal distortion rate. In *International Conference on Learning Representations (ICLR)*, 2026.

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 *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

A FAST VARIANT: SPEEDUP AND CAP CALIBRATION

Isolated compression-stage speedup. Table 3 reports the speedup of the randomized token-mode SVD in isolation on Mistral at T=1024 with q_{cap}=32, measured on pre-grouped tensors with the allocator already swapped. These are isolated stage timings, not end-to-end production figures. LLaMA-2 gives $5.24\times$ to $9.97\times$ on the same stage. On the same isolated path, the reconstruction-error gap between FlashJoLT and JoLT in the free zone at T=1024 is $|\Delta_K| \leq 0.0093$ and $|\Delta_V| \leq 0.0011$ on Mistral and $|\Delta_K| \leq 0.0089$ and $|\Delta_V| \leq 0.0031$ on LLaMA-2 at 2–3×.

Independent wall-clock timing of the two production compressors over the complete pipeline, including layer grouping, gives end-to-end speedups of $1.04\times$ (cache length 821, C=3.7) and $3.97\times$ (cache length 1024, C=4) on an A100 SXM4 40 GB. The token-mode SVD runs inside both the allocation and the fit stages, and FlashJoLT changes both. No single-stage speedup therefore summarizes the end-to-end gain, and the isolated figures in Table 3 do not transfer to the production path.

Table 3: Isolated speedup of the randomized token-mode SVD stage (Mistral, T=1024, q_{cap}=32).

Target C	2	3	4	5	6	7	8	10
isolated speedup	8.87×	9.30×	5.25×	7.76×	6.87×	13.57×	7.57×	12.86×

Cap calibration. The cap policy of Equation (5) is a no-op for $T \leq 1024$ relative to the production default it replaced (32 for $C \leq 4$, 64 for $C > 4$), so every short-context number in this paper is unchanged by it. The isolated sweep above fixes q_{cap}=32 at every C by design. A calibration sweep on real Mistral KV confirms the policy is safe: for $C \leq 5$ the reconstruction-error gate $\Delta_K \leq 0.025$ passes at every context length from 512 to 8192.

Why the cap helps past the free zone. On LLaMA-2-13B at context length n=1024 and 4×, JoLT reaches $+27.28\%$ perplexity (Table 1) while FlashJoLT reaches $+14.16\%$ (Table 16). To separate the cap from the randomized SVD, we run three arms on the same 150 chunks with three seeds each: FlashJoLT, JoLT, and JoLT with the token rank capped at 32 and nothing else changed. Against the matched uncompressed perplexity of 5.56, the three arms land at $+14.21\%$, $+26.47\%$, and $+14.46\%$. The capped exact solver matches FlashJoLT within 0.25 pp and sits 12 pp from the uncapped one, while reconstruction error is the same in all three (0.12 on keys, 0.08 on values).

The difference is the allocation. Without the cap, the allocator assigns token rank 55 to 81 on keys in six of the fourteen layer groups and drops their residual entirely, spending 18.8 MB on key factors and 23.7 MB on key residual. Under the cap, the same groups spend 7.1 MB on key factors and 34.8 MB on key residual. Value allocation is unchanged. This is the surrogate limitation described in Section 3: on LLaMA-2 keys past the free zone, a key residual at 2 to 4 bits protects perplexity better than the extra rank the same bytes would buy. Because RoPE preserves the Frobenius norm, the surrogate scores pre-RoPE and post-RoPE keys identically, but the rotated error can still matter through how it aligns with the queries. An attention-logit-weighted surrogate would test this hypothesis, and we leave the mechanism open. The randomized SVD adds only seed noise (2.1 pp spread for FlashJoLT against 0.02 pp for the capped exact arm) around the same mean.

B FULL PERPLEXITY GRIDS

This appendix reports the full perplexity grids of JoLT (exact SVD) across context lengths, referenced from Section 4. All cells are corpus-level perplexity over the full 300-chunk evaluation set. $\Delta\%$ is relative to the uncompressed baseline at the same context length. FlashJoLT matches these values within the near-lossless free zone (Appendix A).

On Mistral (Table 4) the degradation past the free zone is gradual at every context length. On LLaMA-2 (Table 5) the step between 4× and 5× is sharp, and perplexity peaks at 7× at n=512 and n=1024 and at 8× at n=2048, then falls through 10×. An allocation log on 10 chunks at n=1024 from 6× to 10× shows only monotone changes: no key cell stores residual bits at any of these targets and key ranks shrink, while value cells move from four residual bits to zero. The allocation trend does

not explain the non-monotone perplexity, which we attribute to a mismatch between the Frobenius surrogate and downstream perplexity (Appendix A).

Cross-family degradation detail. Above the free zone, the degradation pattern is model-specific. The corpus-level $8 \times /3\times$ perplexity multiplier at $n{=}1024$ measures the steepness of high-ratio degradation. Mistral is the mildest at $1.24\times$ and Qwen intermediate at $1.70\times$, while both LLaMA models are steep at $2.01\times$ (LLaMA-2) and $2.31\times$ (LLaMA-3.1). Mistral and LLaMA-2 use JoLT (Tables 4 and 5). LLaMA-3.1 and Qwen use FlashJoLT, with the $8\times$ values given under Table 1. This ordering does not track attention type or KV head count: LLaMA-3.1-8B, a GQA model with the same 8 KV heads as Mistral, degrades as steeply as LLaMA-2-13B (MHA, 40 KV heads), while Qwen (also GQA, 8 KV heads) falls between the two. Any operating point above $3\times$ therefore requires per-model validation rather than an architecture rule.

OLMoE (Mixture-of-Experts). OLMoE-1B-7B-0924 (Muennighoff et al., 2025) is a sparse MoE whose sparsity is FFN-only: the attention projections are standard dense layers, so the per-layer KV cache is a standard tensor and JoLT applies without modification. Table 6 reports perplexity against the uncompressed baseline, both as 3-seed means. At $2\times$ the cost is $+0.02\%$ at both context lengths, and at $3\times$ it is $+0.11\%$ to $+0.14\%$. At $4\times$ the drift is a graceful $+3.0\%$ to $+3.2\%$ with no cliff. The seed spread is tight (per-seed $2\times$ deltas span $+0.005\%$ to $+0.045\%$), so near-losslessness is not a single-seed artifact.

Table 4: JoLT perplexity on Mistral-7B-v0.3 (GQA) across context lengths; $\Delta\%$ relative to the uncompressed baseline.

Ratio	$n{=}512$	$n{=}1024$	$n{=}2048$
baseline	6.96	6.28	6.01
$2\times$	6.96 ($+0.07\%$)	6.28 (-0.02%)	6.01 ($+0.01\%$)
$3\times$	6.97 ($+0.11\%$)	6.29 ($+0.04\%$)	6.01 ($+0.07\%$)
$4\times$	7.26 ($+4.37\%$)	6.64 ($+5.59\%$)	6.28 ($+4.63\%$)
$5\times$	7.78 ($+11.81\%$)	7.01 ($+11.56\%$)	6.53 ($+8.76\%$)
$8\times$	8.54 ($+22.69\%$)	7.78 ($+23.77\%$)	7.18 ($+19.48\%$)
$10\times$	8.93 ($+28.36\%$)	8.00 ($+27.29\%$)	7.46 ($+24.17\%$)

Table 5: JoLT perplexity on LLaMA-2-13B (MHA) across context lengths; $\Delta\%$ relative to the uncompressed baseline.

Ratio	$n{=}512$	$n{=}1024$	$n{=}2048$
baseline	5.95	5.39	5.19
$2\times$	5.95 ($+0.04\%$)	5.39 (-0.02%)	5.19 ($+0.01\%$)
$3\times$	5.96 ($+0.08\%$)	5.39 (-0.01%)	5.19 ($+0.05\%$)
$4\times$	6.78 ($+13.94\%$)	6.86 ($+27.28\%$)	6.05 ($+16.72\%$)
$5\times$	10.76 ($+80.70\%$)	9.07 ($+68.29\%$)	7.43 ($+43.26\%$)
$7\times$	13.28 ($+123.10\%$)	11.04 ($+104.93\%$)	8.62 ($+66.31\%$)
$8\times$	12.72 ($+113.64\%$)	10.83 ($+100.90\%$)	8.72 ($+68.21\%$)
$10\times$	11.34 ($+90.47\%$)	9.04 ($+67.82\%$)	7.64 ($+47.28\%$)

Table 6: JoLT perplexity on OLMoE-1B-7B-0924 (MoE) across context lengths, mean over 3 seeds; WikiText-2 only, 100 chunks per seed, a different chunk subset per seed; $\Delta\%$ relative to the uncompressed baseline.

Ratio	n=512	n=1024
baseline	7.9414	7.2635
2×	7.9431 (+0.02%)	7.2652 (+0.02%)
3×	7.9527 (+0.14%)	7.2711 (+0.11%)
4×	8.1945 (+3.19%)	7.4812 (+3.00%)

C DOWNSTREAM DETAILS

This appendix collects the downstream task details referenced from Section 4.3.

GSM8K evaluation. Table 7 reports JoLT's GSM8K accuracy on Mistral-7B and LLaMA-2-13B at its own compression ratios. On both models, 2× and 3× are within noise of the uncompressed baseline, with confidence intervals that include zero. At 4×, Mistral drops by 8.49 pp, consistent with the perplexity cliff past the free zone. LLaMA-2 at 4× degrades more gently (−2.20 pp), with the confidence interval still touching zero.

Table 7: GSM8K accuracy (%), 8-shot chain-of-thought, exact match, JoLT versus the uncompressed baseline (n=1319 per cell). Mistral 2× and 3× are identical across seeds $\{0, 1, 2\}$, all other cells are seed 0.

Model	ratio	accuracy	Δ (pp)	95% CI (pp)
Mistral-7B (GQA)	baseline	40.56		
	2×	40.11	−0.45	[−1.52, +0.61]
	3×	39.35	−1.21	[−2.65, +0.15]
	4×	32.07	−8.49	[−11.1, −5.9]
LLaMA-2-13B (MHA)	baseline	28.13		
	2×	28.43	+0.30	[−0.8, +1.4]
	3×	28.51	+0.38	[−0.9, +1.7]
	4×	25.93	−2.20	[−4.40, +0.08]

GSM8K matched-memory comparison. Table 8 compares JoLT against fixed-rate quantizers at the byte rate of int4 per-channel quantization, roughly 4×, on Mistral-7B. At this rate the quantizers are competitive: KIVI 2-bit scores 40.18% (+3.11 pp over int4's 37.07%) at a slightly looser budget, and TurboQuant scores 38.89% (+1.82 pp). JoLT at the same budget achieves 35.94% at an achieved ratio of 4.1× (−1.14 pp versus int4). The int4-native byte rate forces JoLT past its free zone at short context. At 64K and above, where the KV cache is the actual bottleneck, FlashJoLT costs under one point at 4× (Section 4.3).

Table 8: GSM8K accuracy on Mistral-7B at matched memory (n=1319). Top: JoLT at its own ratios, Δ versus the uncompressed baseline. Bottom: methods near the int4 per-channel byte rate, Δ versus int4. B/token is serialized prefill-KV bytes per token. KIVI and TurboQuant sit slightly above int4's bytes. Deltas are computed from unrounded accuracies.

Method	B/token	ratio	acc (%)	Δ (pp)	
JoLT own ratio (Δ vs uncompressed)					
uncompressed	131072.0	1.0×	40.56		
JoLT 2×	65540.4	2.0×	40.11	−0.45	
JoLT 3×	43673.4	3.0×	39.35	−1.21	
Near int4 bytes (Δ vs int4)					
KIVI 2-bit	35975.6	3.6×	40.18	+3.11	⋆
TurboQuant	34304.0	3.8×	38.89	+1.82	
int4 per-ch.	33089.6	4.0×	37.07	ref	ref
JoLT	32066.5	4.1×	35.94	−1.14	

⋆ KIVI sits at a looser budget.

RULER full grid. Table 9 reports the full RULER grid across context lengths and compression ratios. On Mistral-7B at 4K and 8K (3 dataset seeds, 1200 rows per arm), the 2× and 3× arms track the baseline closely: at 8K the partial-credit deltas are −0.20 pp and −0.70 pp on a 97.37 baseline. Extending to 16K and 32K (2 dataset seeds, 800 rows per arm), both arms remain near-lossless, with pooled deltas of −0.45 pp at 2× and −0.225 pp at 3×. A single-seed 4× stress arm on Mistral shows the penalty shrinking with context: −6.10 pp at 4K, −0.60 pp at 8K, −0.30 pp at 16K, and −0.20 pp at 32K.

Table 9: RULER partial-credit accuracy on the 4-task subset (Hsieh et al., 2024), FlashJoLT. LLaMA-3.1-8B at 64K: 3 dataset seeds, 600 rows per arm. Mistral-7B: 3 seeds at 4K and 8K, 2 seeds at 16K and 32K. Mistral 4× is a single-seed stress arm (seed 42), reported against its own seed's baseline. No 5× arm was run on Mistral. LLaMA-3.1-8B at 4K and 8K was run at 5× only, on an RTX 3090.

Model	context	baseline	2×	3×	4×	5×
LLaMA-3.1-8B (GQA), 64K, 3 dataset seeds						
	65536	90.97	91.20 (+0.23)	90.87 (−0.10)	90.07 (−0.90)	88.57 (−2.40)
Mistral-7B (GQA), 4K to 32K						
	4096	98.80	98.80 (+0.00)	98.80 (+0.00)	92.70 (−6.10)	–
	8192	97.37	97.17 (−0.20)	96.67 (−0.70)	96.77 (−0.60)	–
	16384	95.80	95.60 (−0.20)	95.55 (−0.25)	95.50 (−0.30)	–
	32768	85.90	85.20 (−0.70)	85.70 (−0.20)	85.70 (−0.20)	–
LLaMA-3.1-8B (GQA), 4K and 8K, 3 dataset seeds						
	4096	99.67	–	–	–	92.83 (−6.83)
	8192	99.67	–	–	–	98.53 (−1.13)

LongBench 16K on Mistral-7B. Table 10 reports per-task results for FlashJoLT versus the uncompressed baseline on six LongBench tasks at 16K context (200 examples per task, seed 0, official LongBench metrics).

Table 10: LongBench 16K, Mistral-7B, FlashJoLT versus baseline, per-task means with Δ against the uncompressed baseline in parentheses. Metrics: F1 for HotpotQA and 2WikiMQA, ROUGE-L for GovReport, QMSum and SAMSum, classification accuracy for TREC.

task	baseline	2×	3×	4×
hotpotqa (F1)	12.83	12.72 (−0.11)	12.94 (+0.10)	12.21 (−0.63)
2wikimqa (F1)	11.77	11.88 (+0.11)	11.62 (−0.15)	11.62 (−0.15)
gov_report (RG-L)	28.31	28.24 (−0.07)	29.08 (+0.77)	27.40 (−0.91)
qmsum (RG-L)	22.88	22.90 (+0.02)	22.97 (+0.09)	22.56 (−0.31)
samsum (RG-L)	39.87	39.41 (−0.46)	39.36 (−0.51)	39.24 (−0.63)
trec (acc)	75.50	75.50 (+0.00)	75.50 (+0.00)	75.00 (−0.50)
pooled mean	31.86	31.77 (−0.09)	31.91 (+0.05)	31.34 (−0.52)

D FULL ABLATION GRIDS

This appendix reports the context-resolved ablation grids referenced from Section 4.4. Ablation perplexities are evaluated on a 100-chunk subset (51,100 evaluated tokens per cell) and are comparable within this appendix but not with the full-corpus values of Table 1.

Table 11 collects the headline cells at n=1024 (mean PPL over 3 seeds). Tables 12 and 13 resolve the allocation-strategy and per-group-rank ablations by context length. The joint column of Table 12 and the per-group column of Table 13 are separate runs of the same configuration and agree to within 0.001 PPL (8.145 against 8.144 at n=1024, 8×).

In the allocation-strategy grid (Table 12), the joint and K/V-decoupled solvers track each other closely at 2× across all three context lengths. The joint solver pulls ahead at 8× where the tighter budget makes K/V rebalancing valuable, except at n=2048 where the two stay within 0.17 PPL. Per-group rank allocation (Table 13) beats uniform rank at every cell, with the largest gap at short context and high compression (0.97 PPL at n=512, 4×).

Table 11: Ablations on Mistral-7B-v0.3 at n=1024, mean PPL over 3 seeds. Allocation strategy: joint, K/V-decoupled, and rank-only (no residual). Per-group versus uniform rank. Residual bits at 2×.

	2×	4×	8×
Allocation strategy			
Joint (ours)	6.54	6.93	8.14
K/V-decoupled	6.54	6.72	8.42
Rank-only (no residual)	7.27	7.95	8.65
Per-group vs. uniform rank			
Per-group (ours)	6.54	6.93	8.14
Uniform rank	6.70	7.47	8.68
Residual bits at 2× (0 / 2 / 4 / 8 bit)			
PPL	7.27 / 6.63 / 6.537 / 6.536		

Table 12: Allocation strategy by context length (joint / K/V-decoupled / rank-only), mean PPL over 3 seeds.

n, ratio	joint	K/V-decoupled	rank-only
512, 2×	6.532	6.534	7.412
512, 4×	6.777	6.699	8.121
512, 8×	7.917	8.362	8.463
1024, 2×	6.536	6.535	7.269
1024, 4×	6.928	6.717	7.946
1024, 8×	8.145	8.418	8.649
2048, 2×	5.981	5.981	6.394
2048, 4×	6.259	6.091	6.855
2048, 8×	7.102	7.057	7.388

Table 13: Per-group versus uniform rank by context length, mean PPL over 3 seeds.

n, ratio	per-group	uniform
512, 2×	6.532	6.843
512, 4×	6.777	7.749
512, 8×	7.917	8.712
1024, 2×	6.536	6.702
1024, 4×	6.927	7.470
1024, 8×	8.144	8.680
2048, 2×	5.981	6.059
2048, 4×	6.260	6.509
2048, 8×	7.102	7.185

E CROSS-METHOD RECONSTRUCTION AND PERPLEXITY BENCHMARK

Baseline implementations. KIVI and xKV run the authors' released code, and ReCalKV runs the authors' head-reordering code. Palu, KQ-SVD, and TurboQuant are our reimplementations from the papers, applied to the prompt's cache under the deviations listed below, so their numbers are not reproductions of the originally reported results.

The Palu arm applies Palu's grouped low-rank decomposition to the prompt's cache (a per-prompt SVD of the grouped cache rather than an offline factorization of the projection weights) with uniform per-group rank, in place of Palu's offline Fisher-information rank search. The ReCalKV arm keeps ReCalKV's head reordering and grouped SVD for keys but applies a per-prompt SVD to the value cache in place of offline value calibration, with uniform rank in place of Fisher-based allocation. The KQ-SVD arm computes its key factorization from the prompt's own queries rather than offline calibration queries and uses per-head SVD for values. The TurboQuant arm uses uniform 4 bits with no outlier-channel split, the inner-product variant for keys and the MSE variant for values. Palu is evaluated as an fp16 low-rank method without optional latent quantization. Quantizer key errors are measured on post-RoPE keys and low-rank key errors on pre-RoPE keys.

Reconstruction error. Tables 14 and 15 report mean relative Frobenius reconstruction error at $T{=}1024$ for all methods on each model at $2\times$ and $4\times$.

At $2\times$, JoLT and FlashJoLT rank first and second among the low-rank methods on keys and values on both architectures. The fixed-rate int8 baseline has lower key error (0.0047 on Mistral) but higher value error. On Mistral values, JoLT reaches 0.00571, roughly $27\times$ below Tucker without residual at 0.15546, the best non-JoLT low-rank result. The residual rotation is fixed across seeds, so every method varies across seeds only through the chunk draw. The standard deviations shown for JoLT and FlashJoLT are at most 0.0003, against a gap of at least 0.088 to the nearest baseline.

At $4\times$, keys show a crossover: int4 per-channel outperforms JoLT on Mistral (0.080 against 0.118) and LLaMA-2 (0.077 against 0.120), while JoLT remains better on values (0.090 against 0.132 and

0.087 against 0.124). This does not affect the free-zone claim because the evaluated low-bit fixed-rate quantizers cannot operate in the 2–3× band, but at 4× the best method depends on whether key or value error dominates task quality. Per-head SVD and KQ-SVD have bit-identical value errors because they share a value path. Mistral int4 results use the source's three-decimal precision.

Table 14: Relative Frobenius reconstruction error on Mistral-7B-v0.3 (GQA) at T=1024, mean over 3 seeds; lower is better. Parentheses: across-seed standard deviation at 2× for JoLT and FlashJoLT; the seed changes the reconstruction chunks drawn, not the residual rotation. int4 per-channel has no 2× operating point.

	2×		4×	
Method	K	V	K	V
JoLT (ours)	0.00903 (.0002)	0.00571 (.0001)	0.11844	0.08964
FlashJoLT (ours)	0.00782 (.0003)	0.00584 (.0001)	0.11765	0.09307
xKV (Chang et al., 2026)	0.10185	0.20863	0.16212	0.37706
Palu (Chang et al., 2025)	0.09912	0.31842	0.17219	0.49365
ReCalKV (Yan et al., 2025)	0.09731	0.27728	0.16940	0.44446
Tucker-only (ours, no residual)	0.14027	0.15546	0.20661	0.32525
Per-head SVD	0.13850	0.42096	0.24208	0.60426
KQ-SVD (Lesens et al., 2026)	0.15496	0.42096	0.26287	0.60426
Fixed-rate quantizer at native ≈ 3.97×				
int4 per-channel	n/a	n/a	0.080	0.132

Table 15: Relative Frobenius reconstruction error on LLaMA-2-13B (MHA) at T=1024. FlashJoLT and xKV are 3-seed means, the other low-rank baselines single-seed. JoLT and Tucker-only were not run on this grid. Quantizers at their single evaluated setting.

	2×		4×	
Method	K	V	K	V
FlashJoLT (ours)	0.007452	0.005614	0.120333	0.086566
xKV (Chang et al., 2026)	0.114161	0.190025	0.177292	0.363146
Palu (Chang et al., 2025)	0.116724	0.318463	0.189522	0.478121
ReCalKV (Yan et al., 2025)	0.114969	0.224396	0.186213	0.380975
Per-head SVD	0.148635	0.402574	0.241318	0.574452
KQ-SVD (Lesens et al., 2026)	0.166458	0.402574	0.263680	0.574452
Fixed-rate quantizers at their evaluated setting				
int4 per-channel (≈ 3.97×)	n/a	n/a	0.077078	0.123518
KIVI 2-bit (Liu et al., 2024) (4.20×)	n/a	n/a	0.200598	0.383170
TurboQuant (Zandieh et al., 2026) (3.82×)	n/a	n/a	0.227269	0.096594

Perplexity on the same grid. Table 2 in the main text reports the cross-method perplexity grid on Mistral. Two baselines that do not appear in that table are Tucker-only (ours, no residual), which reaches +10.81% at 2× and +21.23% at 4×, and Per-head SVD, which reaches +8.39% and +15.74% at the same ratios.

Table 16 reports the corresponding grid on LLaMA-2-13B. All LLaMA-2 methods except Palu, ReCalKV, Per-head SVD, and KQ-SVD are 3-seed means. JoLT and Tucker-only were not run on this grid, so FlashJoLT represents the JoLT family. Its +14.16% at 4×, rather than JoLT's +27.28% in Table 1, reflects the token-rank cap of Section 3.1, which binds on LLaMA-2 beyond the free zone (Appendix A). The pattern matches Mistral: JoLT's advantage lies in the 2–3× band where no fixed-rate quantizer operates, and near 4× TurboQuant and KIVI outperform it on perplexity.

At 2×, fixed-rate 8-bit baselines match free-zone quality on both models. int8 per-channel achieves relative Frobenius error 0.0047/0.0077 (K/V) on Mistral at T=1024 (1.99× achieved)

and 0.0045/0.0073 on LLaMA-2 ($1.99\times$), with perplexity within 0.02% of the uncompressed baseline on both models. fp8 e4m3 sits at exactly $2.00\times$ with reconstruction error roughly $6\times$ higher than int8 on keys and $3.5\times$ higher on values, but perplexity remains within 0.04%. int8 per-channel stores one scale per row, so its ratio is $2T/(T+4) < 2$. Neither format has an operating point above $2\times$.

Table 16: Cross-method perplexity on LLaMA-2-13B (MHA) at n=1024 (WikiText-2 and C4, 150 chunks, mean over seeds). Parentheses give the $\Delta\%$ against the grid's uncompressed perplexity of 5.56. JoLT and Tucker-only were not run on this grid. Quantizers run only at their single evaluated setting, shown next to the name.

Method	2×	3×	4×
FlashJoLT (ours)	5.56 (+0.01)	5.56 (+0.03)	6.35 (+14.16)
xKV (Chang et al., 2026)	8.74 (+57.21)	7.79 (+40.14)	7.71 (+38.66)
Palu (Chang et al., 2025)	6.28 (+12.96)	6.55 (+17.76)	6.75 (+21.48)
ReCalKV (Yan et al., 2025)	6.81 (+22.59)	7.02 (+26.35)	7.21 (+29.71)
Per-head SVD	6.01 (+8.04)	6.24 (+12.22)	6.45 (+16.12)
KQ-SVD (Lesens et al., 2026)	5.87 (+5.52)	6.04 (+8.67)	6.22 (+11.99)
Fixed-rate quantizers at their evaluated setting (single operating point, shown in the 4× column)			
KIVI 2-bit (Liu et al., 2024) (≈ 4.20×)	–	–	5.65 (+1.59)
TurboQuant (Zandieh et al., 2026) (≈ 3.82×)	–	–	5.57 (+0.25)
int4 per-channel (≈ 3.97×)	–	–	6.01 (+8.08)
Fixed-rate 8-bit at native rate (shown in the 2× column)			
int8 per-channel (≈ 1.99×)	5.56 (+0.02)	–	–
fp8 e4m3 (2.00×)	5.56 (+0.04)	–	–