OSCAR: Offline Spectral Covariance-Aware Rotation for 2-bit KV Cache Quantization

Zhongzhu Zhou[*,1,2] **Donglin Zhuang**[*,1,2] **Jisen Li**[1,3] **Ziyan Chen**[2]
Shuaiwen Leon Song[1] Ben Athiwaratkun[1] Xiaoxia Wu[†,1]

[1]Together AI [2]University of Sydney [3]University of Illinois Urbana-Champaign

[*]Equal contribution. [†]Corresponding author.

Abstract

INT2 KV-cache quantization is attractive for long-context LLM serving, yet making it both accurate and practical remains challenging. Data-oblivious rotations such as Hadamard transforms flatten channel outliers, but still degrade sharply at INT2 because they ignore how downstream attention queries the cache. We propose OSCAR, an ultra-low-bit KV-cache quantization method that estimates *attention-aware covariance* targets offline and uses them to derive fixed per-layer rotations and clipping thresholds. By aligning the quantization basis with the second-moment structures consumed by attention, OSCAR preserves attention fidelity at 2-bit precision. Beyond theoretical analysis under a frozen-error surrogate, we build a serving-ready OSCAR system with fused Triton write and decode kernels that preserve paged KV-cache layouts and prefix caching, integrating into modern serving engines such as SGLang and vLLM.

We evaluate OSCAR on reasoning and coding benchmarks with generation traces of up to 32K tokens across five tasks. On Qwen3-4B-Thinking-2507 and Qwen3-8B, OSCAR narrows the BF16 accuracy gap to 3.78 and 1.21 points, respectively, where Hadamard-only INT2 collapses, and matches BF16 within run-to-run variance on Qwen3-32B and GLM-4.7 (358B parameters). On RULER-NIAH up to 128K tokens, OSCAR substantially outperforms QuaRot-style INT2 across context lengths. System-wise, OSCAR compresses KV-cache memory by nearly $8\times$, boosts multi-request throughput by up to $7.8\times$ at large batch sizes within the same memory budget, and accelerates batch-size-1 decoding by up to $3.1\times$ over BF16 by cutting KV memory traffic.

Code | Website | RotationZoo

1 Introduction

Long-context inference has made the key–value (KV) cache one of the primary bottlenecks in serving large language models. During autoregressive decoding, the cache grows linearly with context length, batch size, and model depth, and every generated token must read cached keys and values from GPU memory. Compressing the KV cache is therefore a direct way to support larger batch sizes and reduce memory bandwidth traffic [1–8]. Among the available design points, token-wise INT2 quantization offers a compelling system trade-off: it provides a large memory reduction while retaining a regular, fixed-width layout that maps cleanly onto paged attention kernels.

Aggressively compressing KV caches to ultra-low precision (e.g., INT2) remains difficult because KV activations exhibit severe channel-wise outliers: a small subset of channels carries large magnitudes while most channels remain modest [4]. Under 2-bit quantization, these outliers stretch the quantization scale and collapse normal values onto one or two quantization levels, degrading

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

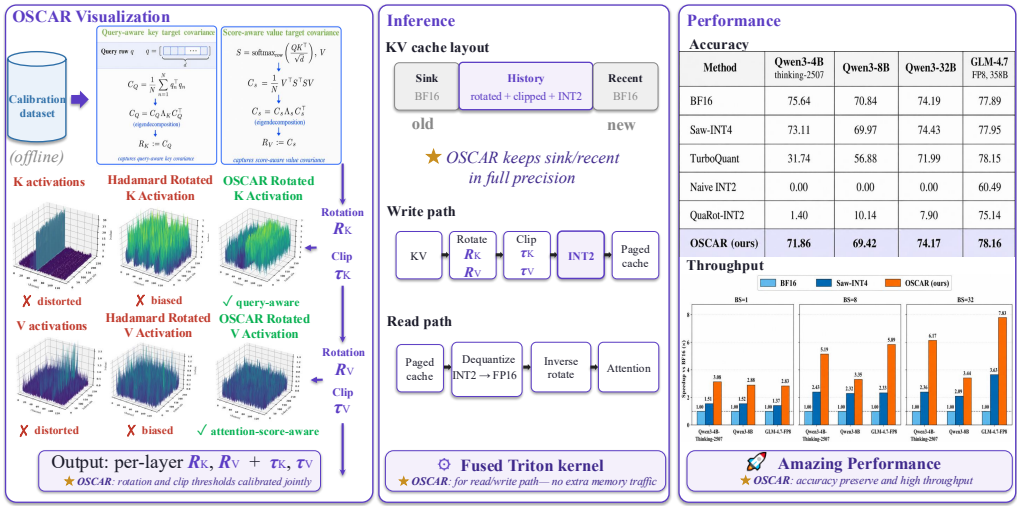

Figure 1: **OSCAR pipeline overview.** Offline, OSCAR estimates attention-aware key/value covariance rotation and shows how the resulting rotation makes KV activations more uniform: Hadamard mixing flattens raw peaks, while the OSCAR rotation separates directions that matter more or less to attention. Online, the serving path keeps sink and recent tokens in BF16 while applying the fixed rotate–clip–INT2 path to history KV tokens inside the paged SGLang cache. The right panels summarize that OSCAR preserves accuracy across models while achieving throughput close to the QuaRot(INT2) serving path and higher throughput than BF16.

attention quality. Rotation-based quantization mitigates this issue by applying a fixed orthogonal transform, such as a Hadamard rotation, to spread extreme values across channels and produce a more uniform dynamic range [9–12]. Because an orthogonal rotation preserves tensor shapes and requires no per-channel routing or irregular sparse metadata, it remains naturally compatible with paged KV-cache layouts [13] and FlashAttention-style fused decode kernels [14–17]: each KV vector is rotated into a better-conditioned basis before quantization and rotated back during attention [12].

However, a data-oblivious rotation only equalizes channel magnitudes without considering which feature directions matter to attention. At INT2, where only four quantization levels are available, this distinction is critical: quantization error along eigenvectors heavily queried by Q perturbs attention logits far more than error along weakly queried directions. Because attention consumes keys through query inner products and values through score-weighted sums rather than raw Euclidean reconstruction, the rotation basis should be derived from attention statistics themselves. Motivated by this observation, we propose OSCAR, an INT2 KV quantizer that estimates attention-aware covariance targets from a lightweight offline calibration pass and uses them to construct separate key and value rotations alongside per-token clipping thresholds. Figure 1 (left) illustrates that while Hadamard rotation partially flattens raw channel spikes, combining attention-aware covariance alignment with Hadamard mixing produces a much smoother activation distribution for 2-bit quantization.

Our contributions are summarized as follows

- **We identify the missing target in rotation-based INT2 KV quantization.** Data-oblivious rotations flatten raw activation outliers, but INT2 generation quality depends on preserving attention logits and score-weighted value outputs; consequently, the rotation basis should be derived from downstream attention statistics rather than raw cache reconstruction.
- **We propose OSCAR, an attention-aware calibration framework for 2-bit KV caches.** From a single offline calibration pass, OSCAR estimates query- and score-aware second-moment targets for keys and values and composes their eigenbases with Hadamard mixing and bit-reversal grouping. We analyze these targets under a frozen-error surrogate objective and evaluate OSCAR across models from 4B to 358B parameters, retaining near-BF16 reasoning and coding accuracy at 2.28 bits per KV element.
- **We build a serving-ready INT2 KV-cache system.** OSCAR stores historical KV tokens in a uniform rotated INT2 paged layout while keeping attention sinks and a sliding recent window in BF16 [18–21]. Integrated into SGLang and vLLM [22, 23] with custom fused Triton write and

decode kernels that preserve prefix caching [24], OSCAR delivers up to 7.8× higher job throughput and 3.1× faster batch-1 decoding at 100K context length.

Due to space constraints, we provide a full discussion in Appendices C and D on how our design builds on and differs from prior work.

2 Preliminaries and Motivation

Attention and KV cache. We use *row-vector* notation throughout. For simplicity, we define single-head attention [25] as follows. Given a sequence of T hidden states $\{x_t\}_{t=1}^{T}$ with $x_t \in \mathbb{R}^{1\times d}$ and projection weights $W_Q, W_K, W_V \in \mathbb{R}^{d\times d}$, the query, key, and value matrices are $Q = [q_1; \ldots; q_T] \in \mathbb{R}^{T\times d}$, $K = [k_1; \ldots; k_T] \in \mathbb{R}^{T\times d}$, $V = [v_1; \ldots; v_T] \in \mathbb{R}^{T\times d}$, where d is the head dimension and $q_t = x_t W_Q \in \mathbb{R}^{1\times d}$, $k_t = x_t W_K \in \mathbb{R}^{1\times d}$, and $v_t = x_t W_V \in \mathbb{R}^{1\times d}$. The attention score matrix is $S = \mathrm{softmax}_{\mathrm{row}}(QK^\top/\sqrt{d}) \in \mathbb{R}^{T\times T}$, and the attention output is $O = SV \in \mathbb{R}^{T\times d}$, or equivalently $o_i = \sum_{t=1}^{T} S_{i,t} v_t$. During autoregressive inference, $K_{1:t} = [k_1; \ldots; k_t]$ and $V_{1:t} = [v_1; \ldots; v_t]$ are cached across steps as the KV cache. Grouped-query attention (GQA) [26], covariance derivations, and serving procedures are detailed in Appendices A and B.

Quantization notation. A b-bit quantizer consists of a quantization map $\mathcal{Q}^+ : \mathbb{R} \to \mathcal{C}_b$ and a dequantization map $\mathcal{Q}^- : \mathcal{C}_b \to \mathbb{R}$, where $\mathcal{C}_b$ is a discrete set of 2^b representable codes. Their composition gives the quantize–dequantize map $\mathcal{Q}(x) = \mathcal{Q}^-(\mathcal{Q}^+(x))$. A symmetric uniform map can be written as $\mathcal{Q}(x) = s \cdot \mathrm{clip}(\lfloor x/s \rceil, -\tau, \tau)$, where $s > 0$ is the quantization scale and τ is the clipping limit. For a matrix X consisting of row vectors, we apply quantization row-wise.[1] Given an orthogonal rotation matrix R [9], the reconstructed caches are $\widehat{K} = \mathcal{Q}(KR)R^\top$ and $\widehat{V} = \mathcal{Q}(VR)R^\top$.

Why raw-cache reconstruction is not enough. Rotation is effective for KV-cache quantization because it spreads large channel values into a basis with a more uniform dynamic range [9]. However, simple data-oblivious rotations such as Hadamard or random orthogonal transforms are often insufficient: they smooth the cached tensors, but do not identify which directions are most important for downstream attention. A standard tensor-reconstruction view minimizes $\|K - \mathcal{Q}(K)\|_F^2$ and $\|V - \mathcal{Q}(V)\|_F^2$, yet attention does not consume K and V through their Euclidean reconstruction errors. Keys are used through logits, and values are used through the attention-weighted aggregation. For keys K, the downstream logit distortion is

$$\|QK^\top - Q\widehat{K}^\top\|_F^2 = \mathrm{tr}\Big((K - \widehat{K})Q^\top Q(K - \widehat{K})^\top\Big), \tag{1}$$

which is controlled by the query covariance $Q^\top Q$ rather than by $K^\top K$ alone. For values V with quantized $\widehat{V} = \mathcal{Q}(V)$, the downstream output distortion is

$$\|SV - S\widehat{V}\|_F^2 = \mathrm{tr}\Big((V - \widehat{V})^\top S^\top S(V - \widehat{V})\Big), \tag{2}$$

which depends on how attention scores weight the value rows. Thus, if the goal is to reduce empirical attention distortion rather than raw-cache reconstruction error, the rotation should be estimated from target covariance induced by the attention computation itself.

Figure 2 illustrates this gap across all 36 layers of Qwen3-4B-Thinking-2507. Naive INT2, Hadamard-only rotation, and clipping alone still leave substantial attention-score KL divergence and block-output MSE. In contrast, calibrating the rotation from attention-induced target covariances before quantization consistently lowers distortion across all four stages. A concrete numerical walkthrough of each rotation factor is given in Appendix A.4.

3 Algorithm Design: Offline Calibration and Justification

The method has two phases: an offline calibration phase that estimates target covariances, constructs per-layer rotation matrices, and sets per-token clipping thresholds; and an online serving phase that applies these fixed transforms inside a mixed-precision paged cache layout. All target covariances and rotations are estimated once offline from a small calibration set of post-RoPE activations.

[1] Row-wise quantization is used in our theoretical surrogate; in implementation, asymmetric affine INT2 quantization is applied per token over contiguous channel groups of size 128, 64, or 32 (Appendix A.5).

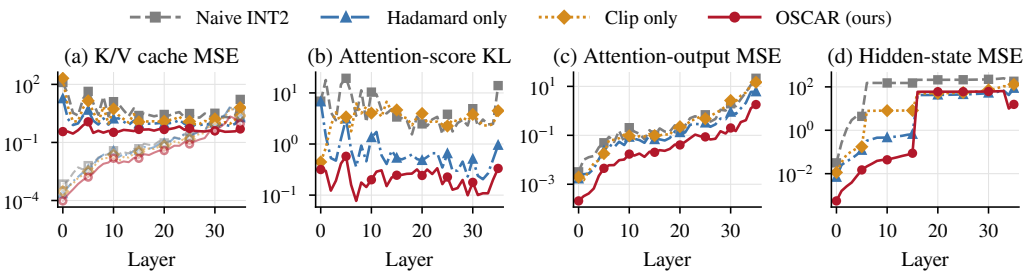

Figure 2: Quantization error. OSCAR limits it at every stage (Qwen3-4B-Thinking-2507, AIME). **(a)** MSE in quantized K and V caches (keys solid, values dashed). **(b)** $\mathrm{KL}(p_{\mathrm{FP16}} \,\|\, p_q)$ between FP16 and quantized attention score distributions. **(c)** MSE at the attention-block output (post-W_O). **(d)** MSE in the propagated hidden states across layers. Curves compare naive INT2, Hadamard-only preprocessing, clip-only outlier control, and full OSCAR.

Query-aware key target covariance. For a query row q, with the attention-aware key target covariance $Q^\top Q$ introduced in equation (1), its empirical estimator is $C_Q = \frac{1}{N}\sum_{n=1}^{N} q_n^\top q_n$; we apply eigen decomposition as in App. A.2, $C_Q = U_Q \Lambda_Q U_Q^\top$, and define the rotation as $R_k := U_Q$.

Score-aware value target covariance. For the attention matrix $S = \mathrm{softmax}_{\mathrm{row}}(QK^\top/\sqrt{d})$, we heuristically define the target covariance in equation (2): $C_S = \frac{1}{N} V^\top S^\top S V$. We then compute the eigendecomposition as in App. A.2, $C_S = U_S \Lambda_S U_S^\top$, and define the raw rotation as $R_v := U_S$.

OSCAR rotations. Following rotation-based low-bit quantization methods [9, 27], we compose the base rotation with a Hadamard transform H_{Had} (App. A.1) and a bit-reversal permutation P_{br} to form the final rotations for keys and values:

$$R_K = U_Q\, H_{\mathrm{Had}}\, P_{\mathrm{br}}, \qquad R_V = U_S\, H_{\mathrm{Had}}\, P_{\mathrm{br}}. \tag{3}$$

Here R_k and R_v are the covariance eigenbases, H_{Had} redistributes channel outlier energy across coordinates, and P_{br} [28, 29] applies a deterministic bit-reversal permutation across channels to balance quantization groups (see Appendix A.4 for an intuitive walkthrough and Appendix E.4 for permutation ablations).

Scale determination and per-token clipping. Our quantization backend follows standard post-training quantization practice. We use affine asymmetric INT2 quantization with scale and zero point for both keys and values, together with percentile-based clipping to control outliers. These mechanisms are common in low-bit activation and KV-cache quantization pipelines [20, 4, 30, 31], with technical details shown in App. A.5.

Optimality of targets under a frozen diagonal residual surrogate. Below we show that under a frozen diagonal residual surrogate, the target covariances $C_Q = \frac{1}{N}\sum_{n=1}^{N} q_n^\top q_n$ and $C_S = \frac{1}{N} V^\top S^\top S V$ minimize the surrogate attention distortion. The proof, the streaming score-weighted estimator used for C_S on long calibration sequences, and an empirical check verifying that the deployed quantization residual is strongly diagonal-dominant on real activations (98.4% diagonal energy for keys and 95.5% for values) are provided in Appendices A.6–A.8.

Theorem 1 (Optimal base rotation under a frozen diagonal residual surrogate)**.** *Consider the frozen-error surrogate objectives for the key and value base rotations:* $\tilde{\mathcal{L}}_K(R_k) = \mathrm{tr}(R_k^\top C_Q R_k E_K), R_k^\top R_k = I_d$ *and* $\tilde{\mathcal{L}}_V(R_v) = \mathrm{tr}\big(R_v^\top C_S R_v E_V\big), R_v^\top R_v = I_d$*, where* $C_Q = U_Q \Lambda_Q U_Q^\top$ *and* $C_S = U_S \Lambda_S U_S^\top$ *have eigenvalues sorted in non-increasing order, and the frozen residual second moments are fixed diagonal matrices* $E_K = \mathrm{diag}(\mu_1, \ldots, \mu_d)$ *with* $\mu_1 \le \cdots \le \mu_d$ *and* $E_V = \mathrm{diag}(\eta_1, \ldots, \eta_d)$ *with* $\eta_1 \le \cdots \le \eta_d$*. Then* $R_k = U_Q$ *and* $R_v = U_S$ *minimize* $\tilde{\mathcal{L}}_K(R_k)$ *and* $\tilde{\mathcal{L}}_V(R_v)$ *over orthogonal rotations.*

4 System Design: Online Serving with 2-bit KV Cache

KV Cache Layout. We integrate OSCAR into the SGLang [24] serving stack as an INT2 KV-cache mode with full compatibility with paged-attention [13]. The implementation preserves two short high-precision windows: the first S_0 tokens, which behave as attention sinks, and the most recent W

tokens before the current position. The rest of the middle context is stored in INT2 after the fixed OSCAR rotation. Thus, at decoding position t, the logical cache consists of

$$\underbrace{[1, S_0]}_{\text{bf16 sink}} \parallel \underbrace{[S_0+1, t-W]}_{\text{int2 history}} \parallel \underbrace{[t-W+1, t]}_{\text{bf16 recent}}$$

KV Cache Update. During prefill, the runtime writes cache rows through a fully fused Triton kernel [32]. For a token with BF16 rows k_t, v_t and per-token clipping thresholds $\tau_t^{(K)}, \tau_t^{(V)}$, the stored INT2 codes are

$$k_t^+ = \mathcal{Q}_2^+(\text{clip}(k_t R_K, \tau_t^{(K)})), \qquad v_t^+ = \mathcal{Q}_2^+(\text{clip}(v_t R_V, \tau_t^{(V)})),$$

with four 2-bit values packed per byte.

New decode tokens are first written to the recent window as $k_t R_K$ and $v_t R_V$. As decoding advances, the oldest recent token is demoted into the INT2 middle region by a fused Triton kernel that applies the same clip–quantize operation. OSCAR further eliminates runtime value-rotation overhead by folding R_V and $R_V^\top$ offline into the projection weights W_V and W_O.

Decoding Attention Kernel. During decoding, OSCAR partitions each request's cache indices into BF16 (sink + recent) and INT2 segments on the GPU. The INT2 kernel unpacks bytes, applies the stored scale/zero parameters, and accumulates them in floating point. Existing decoding attention kernels typically consist of two kernel launches [33], one for parallel processing on KV cache segments along the sequence dimension. The second one is used to merge partial attention results from segments with online softmax [14]. OSCAR introduces an additional kernel for BF16 KV cache attention, then reuses the second merge kernel to piggyback on the merge of high-precision partial results. Since the BF16 (sink + recent) segment has orders of magnitude fewer elements than the INT2 segment, the overhead is negligible.

5 Experiments

5.1 Experimental Setup

Models & Benchmarks. We evaluate OSCAR on four model configurations: **Qwen3-4B-Thinking-2507**, **Qwen3-8B**, **Qwen3-32B**, and **GLM-4.7-FP8** [34, 35]. These models span a compact reasoning model, mid-sized dense models, and a 358B MoE model, testing OSCAR across different scales and INT2 sensitivity regimes. We evaluate accuracy on five reasoning and coding benchmarks: AIME25 [36], GPQA-Diamond [37], HumanEval [38], LiveCodeBench v6 [39], and MATH500 [40, 41].

We also evaluate long-context retrieval with **RULER** [42] to test the long-sequence robustness. Particularly, RULER-NIAH is the cleanest stress test: the answer is explicitly present in the prompt, and the main question is whether quantized history tokens can be attended to in a long context.

Table 1: **AIME25 at 32K generation.**

Method	BPE	Qwen3-8B	Qwen3-32B
Original BF16	16.00	66.00±7.33	72.59±7.41
KIVI-KV2 [20]	2.25	52.33±9.00	57.41±9.26
KIVI-KV2* [20]	2.26	57.67±9.00	59.05±12.38
Kitty [6]	2.39	59.67±10.33	69.26±9.26
OSCAR	2.38	66.67±3.33	74.00±5.48

Hardware and Framework. Qwen3-4B-Thinking and Qwen3-8B are served on a single NVIDIA H100 (80 GB); Qwen3-32B and GLM-4.7-FP8 are served on 2×H100 and 8×H100 with tensor parallelism, respectively. All system-level runs use our SGLang [24] implementation.

Generation Protocol & Calibration. We use temperature T=0.6 (T=1.0), top-p=0.95, and top-k=20 for Qwen3-family (GLM-4.7), with thinking mode enabled for reasoning benchmarks. Unless otherwise noted, each Qwen configuration is run with **5 independent seeds** and GLM-4.7-FP8 with **3 runs**; we report mean ± standard deviation. All accuracy evaluations use a maximum generation length of **32768 tokens** and run end-to-end inside SGLang, using the same execution path as our system measurements. All OSCAR parameters are estimated once from a small MMLU/GPQA-style calibration set. For each model, we run one calibration pass and dump per-layer Q, K, V activations (8878 tokens × number of layers), from which we compute the key/value rotations and calibration-derived clipping ratios, then reuse the same parameters for all benchmarks. No task-specific calibration is used.

Table 2: INT2 KV-cache quantization methods compared on four model configurations and five benchmarks. Entries are $\mu \pm \sigma$ over 5 seeds, except GLM-4.7-FP8 (3 runs)*. BPE denotes effective bits per KV element at 128K context length; "no MP" means mixed precision is disabled. "Drop" is the average-column gap to the BF16 upper bound; smaller is better.

Model	Method	BPE	GPQA	HumanE	LCB v6	AIME25	MATH500	Mean	Drop
Qwen3-4B Thinking-2507	BF16	16.00	67.27±1.80	94.05±0.54	48.66±2.20	74.67±1.83	93.55±0.33	75.64	–
	Saw-INT4 [12]	4.25	66.37±2.19	89.78±0.80	46.20±1.94	70.00±4.08	93.19±0.51	73.11	-2.53
	TurboQuant (no MP) [44]	3.25	41.41	31.83	0.58	16.67	68.20	31.74	-43.90
	QuaRot-INT2 [9]	2.25	0.34±0.48	0.98±0.10	0.00±0.00	0.00±0.00	5.67±0.19	1.40	-74.24
	Naive INT2	2.25	0.00	0.00	0.00	0.00	0.00	0.00	-75.64
	OSCAR (ours)	2.28	64.95±1.16	92.24±1.02	45.38±1.97	64.00±3.65	92.75±0.39	71.86	-3.78
Qwen3 -8B	BF16	16.00	56.67±2.30	85.95±1.01	49.01±2.13	70.00±3.33	92.59±0.62	70.84	–
	Saw-INT4 [12]	4.25	54.85±3.17	86.44±0.42	47.95±2.15	68.00±2.98	92.63±0.27	69.97	-0.87
	TurboQuant (no MP) [44]	3.25	55.05	74.63	21.05	46.67	87.00	56.88	-13.96
	QuaRot-INT2 [9]	2.25	14.98±0.63	9.80±0.76	0.58±0.58	2.22±1.57	23.13±1.88	10.14	-60.70
	Naive INT2	2.25	0.00	0.00	0.00	0.00	0.00	0.00	-70.84
	OSCAR (ours)	2.28	55.05±1.47	87.88±0.44	46.32±1.91	66.67±3.33	92.22±0.83	69.63	-1.21
Qwen3 -32B	BF16	16.00	58.49±1.21	91.19±1.11	59.06±1.80	68.67±5.58	93.55±0.41	74.19	–
	Saw-INT4 [12]	4.25	59.29±0.76	90.85±0.47	56.02±2.41	72.67±4.90	93.31±0.10	74.43	+0.24
	TurboQuant (no MP) [44]	3.25	58.69	88.41	55.56	66.67	90.60	71.99	-2.20
	QuaRot-INT2 [9]	2.25	19.70±1.09	1.83±0.33	0.39±0.55	0.00±0.00	17.60±1.66	7.90	-66.29
	Naive INT2	2.25	0.00	0.00	0.00	0.00	0.00	0.00	-74.19
	OSCAR (ours)	2.28	60.40±3.47	90.12±0.71	53.57±0.67	74.00±5.48	92.75±0.46	74.17	-0.02
GLM-4.7 FP8, 358B	BF16	16.00	73.23±1.33	91.46±0.65	49.12±0.59	80.00±3.33	95.66±0.61	77.89	–
	Saw-INT4 [12]	4.25	73.74±1.01	91.30±0.92	50.49±1.47	78.89±1.92	95.32±0.12	77.95	+0.06
	TurboQuant (no MP) [44]	3.25	66.67	90.24	58.48	80.00	95.39	78.15	+0.26
	QuaRot-INT2 [9]	2.25	68.01±3.79	89.87±1.78	48.54±4.79	78.89±1.92	90.40±0.35	75.14	-2.75
	Naive INT2	2.25	54.55±3.54	86.50±1.43	37.03±6.41	37.78±1.92	86.60±0.80	60.49	-17.40
	OSCAR (ours)	2.28	73.57±0.58	91.06±0.25	52.63±1.01	78.89±1.92	94.66±0.76	78.16	+0.27

*TurboQuant entries are single-run results as its vLLM path is too slow for repeated 32K evaluations under our time budget.

Baselines. We use two baseline groups. (1) *Group A* contains channel-wise KV methods such as KIVI and Kitty. These methods require residual buffers, channel-wise scales, promoted channels, or custom page layouts, and we do not have paged/fused kernels for them at 32K generation length.

For the only shared 32K accuracy setting, we show the reported Qwen3-8B and Qwen3-32B AIME25 results in the Table 1; The BPE values include INT payload, BF16 scale/zero metadata with group size 128, and BF16 initial tokens where applicable; the Kitty row is the 12.5% key-channel boost variant. (2) *Group B* contains rotation-based methods: FP16/BF16, naive INT2/INT4, QuaRot-style Hadamard rotation, block-diagonal Hadamard (Saw-INT4) [12], and TurboQuant [43]. For TurboQuant, we use the official vLLM implementation from PR #38479 [43] and run the same 32K-generation evaluation; for fairness, we disable its mixed precision setup and quantize all layers.

For OSCAR we always pair rotation with sink (S=64) and recent-window (R=256) BF16 protection, calibration-derived clip ratios (c_K=0.96 and c_V=0.92, see the calibration paragraph above), and per-token grouped asymmetric INT2 quantization. In Table 2, BPE counts INT payload plus BF16 scale/zero metadata with block size 128. For OSCAR, we report effective BPE at 128K context length, including the BF16 sink/recent tokens (S=64, R=256).[2] LiveCodeBench v6 is evaluated with the same 32K generation cap as the other tasks; this truncates some long code-generation outputs, so the reported LCB numbers are lower than they would be under a longer 128K generation budget.

5.2 Main Results

Table 2 reports the main accuracy comparison on Qwen3-4B-Thinking-2507, Qwen3-8B, Qwen3-32B, and GLM-4.7-FP8.

For OSCAR we use the configuration selected by the ablation studies in Section 5.3: sink=64, recent=256, calibration-derived clip thresholds, attention-aware key/value covariance, and asymmetric INT2 quantization. The key observation is that OSCAR is the only near-2-bit method that remains close to the BF16 accuracy frontier under the 32K-generation evaluation. On Qwen3-4B-Thinking-2507 and Qwen3-8B, OSCAR reduces the BF16 gap to 3.78 and 1.21 points, respectively, while TurboQuant drops 43.90 and 13.96 points and rotation-only INT2 baselines largely collapse.

[2] Qwen3-4B-Thinking-2507 and GLM-4.7-FP8 use group size G=128 (2.28 BPE at 128K context); Qwen3-8B and Qwen3-32B use G=64 (2.53 BPE at 128K when accounting for finer-grained BF16 scale/zero metadata; full $G \in \{128, 64, 32\}$ runs are reported in Appendix E.2).

On Qwen3-32B and GLM-4.7-FP8, OSCAR is essentially tied with BF16 at 2.28 BPE, while Saw-INT4 uses 4.25 BPE; small per-task differences of 1–2 points in either direction reflect sampling and reasoning-length variation under a fixed 32K cap [45] (further models, tool use, and baseline comparisons are reported in Appendix E).

Long-Context Robustness. Table 3 reports RULER-NIAH accuracy from 4k to 128k tokens on the same serving-compatible baselines used in the main accuracy comparison. The expected pattern is simple: short contexts should remain close to BF16, while longer contexts expose accumulated attention-logit error. OSCAR should degrade more slowly than rotation-only INT2 baselines because its rotation target is chosen from the covariance seen by attention, not from raw-cache reconstruction, and because long contexts amplify accumulated KV quantization error. Figure 3 provides a direct KL-based check before the retrieval results. Across both Qwen3-4B-Thinking-2507 and Qwen3-8B, OSCAR is more stable as sequence length increases: its attention distribution stays closer to FP16, while naive INT2 and Hadamard-only rotation drift more quickly. On Qwen3-4B-Thinking-2507, QuaRot-INT2 is already near zero at short contexts, whereas OSCAR stays close to BF16 through 16k and retains non-trivial retrieval at 64k and 128k. On Qwen3-8B, QuaRot-INT2 remains usable only at 4k–8k and collapses after 16k, while OSCAR keeps substantially higher retrieval accuracy throughout the sweep. The GLM-4.7-FP8 preliminary run shows a complementary large-model case: all three methods remain strong on this retrieval-only task, and OSCAR matches the BF16 curve up to 128k. Together, the RULER results and KL curves indicate that attention-aware covariance calibration mainly helps when long histories make small KV errors accumulate over many steps.

Table 3: Long-context retrieval accuracy on **RULER-NIAH** [42] across context lengths from 4k to 128k tokens, on the Group B serving-compatible baselines. Each number is averaged over 3 random seeds, except GLM-4.7-FP8 preliminary results (one run, as GLM-4.7 is very stable).

Model	Method	4k	8k	16k	32k	64k	128k
Qwen3-4B (-Thinking-2507)	BF16	100.0±0.0	100.0±0.0	99.7±2.0	99.3±1.0	85.3±5.0	81.0±1.4
	QuaRot-INT2 [9]	0.0±0.0	0.8±0.2	0.0±0.0	0.0±0.0	15.6±3.2	0.0±0.0
	OSCAR	99.7±0.1	100.0±0.0	97.8±0.2	87.6±0.1	61.9±1.3	39.5±1.0
Qwen3-8B	BF16	100.0±0.0	99.6±0.0	98.9±0.1	97.3±0.0	79.2±0.1	78.2±1.0
	QuaRot-INT2 [9]	74.0±1.0	80.6±0.3	19.0±2.0	9.8±1.0	0.0±0.0	0.0±0.0
	OSCAR	99.5±0.2	97.8±0.3	93.9±0.7	86.3±0.6	61.9±1.8	45.0±0.0
GLM-4.7-FP8†	BF16	100.0	100.0	100.0	100.0	98.8	97.2
	QuaRot-INT2 [9]	100.0	99.7	100.0	100.0	98.8	96.3
	OSCAR	100.0	100.0	100.0	100.0	98.8	97.2

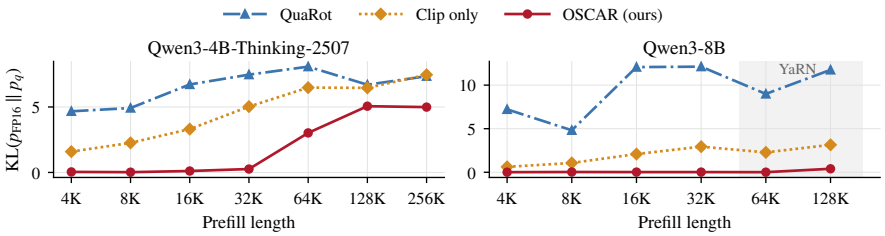

Figure 3: **OSCAR keeps attention distributions stable as context length grows.** KL divergence against the FP16 attention distribution across context lengths for clip-only INT2, QuaRot (Hadamard), and OSCAR on Qwen3-4B-Thinking-2507 and Qwen3-8B.

5.3 Ablation Studies

Rotation Analysis: Decomposing R and Comparing Rotation Targets. OSCAR's composed rotation is $R = U \cdot H_{\text{Had}} \cdot P_{\text{br}}$, the product of an attention-aware eigenbasis U, a Hadamard transform H_{Had}, and a bit-reversal permutation P_{br}. Table 4 removes each factor of R in turn (*top block*) and, on the same footing, changes the PCA target used to compute U (*bottom block*): OSCAR's attention-aware targets are replaced by raw-cache reconstruction targets ($K^\top K / V^\top V$), fixed Hadamard rotations, or no learned rotation.

Two observations emerge. First, both the attention-aware eigenbasis U and the Hadamard component H_{Had} contribute substantially; the bit-reversal permutation P_{br} does not change accuracy in floating-

point math but improves quantization geometry by interleaving large and small eigenvalues so that per-group quantization sees a more uniform range. Second, none of the alternative rotation targets (matched Hadamard $H_{\text{Had}} \cdot P_{\text{br}}$, raw $K^\top K / V^\top V$ reconstruction targets, or no rotation) matches the attention-aware C_Q/C_S targets at the same INT2 budget. This isolates the central claim of the paper: *which* covariance matrix one diagonalizes matters more than *whether* one diagonalizes at all.

Sink and Recent Window Sizes. We sweep $(S, R) \in \{(0,0), (32,128), (64,256), (128,512), (256,1024)\}$ on Qwen3-4B-Thinking-2507 and report accuracy with the additional BF16 KV memory that each protection window introduces (Table 5). A clear knee emerges at (S, R)=$(64, 256)$: smaller windows leave noticeable accuracy on the table, while larger windows provide negligible additional accuracy at the cost of substantially more BF16 KV memory. $(64, 256)$ is a sweet spot and thus used for the main results.

Table 4: Decomposition of the composed OSCAR rotation $R = U \cdot H_{\text{Had}} \cdot P_{\text{br}}$ (*top block*) and comparison against alternative rotation-target choices used by prior work (*bottom block*), all at the same INT2 bit budget with sink/recent protection on Qwen3-8B. Entries are mean ± std over 3 runs.

Configuration	GPQA	HumanE	LCB v6	AIME25	MATH500	Mean
FP16 reference	57.07±0.94	84.71±0.99	48.07±2.20	66.00±2.79	88.28±0.46	68.83
Decomposition of OSCAR's composed rotation						
Full OSCAR: $U \cdot H_{\text{Had}} \cdot P_{\text{br}}$	55.89±1.27	87.97±0.19	46.00±1.88	67.78±1.92	92.38±0.20	70.01
w/o P_{br}	52.86±1.62	86.71±0.84	45.22±0.34	63.33±8.82	91.85±1.45	68.00
w/o H_{Had} (only $U \cdot P_{\text{br}}$)	51.35±1.91	83.78±0.37	19.49±2.05	21.11±1.92	82.97±0.80	51.74
Alternative rotation-target choices						
Matched Hadamard baseline ($H_{\text{Had}} \cdot P_{\text{br}}$, w/o U)	40.74±3.44	38.54±1.18	4.48±0.68	15.56±1.92	64.80±1.29	32.82
OSCAR with $K^\top K / V^\top V$ PCA target	38.72±1.27	35.53±0.67	3.70±0.68	13.33±3.33	64.33±0.53	31.12
no rotation (only clip + sink + recent)	18.52±3.29	0.00	0.39±0.68	2.22±1.92	0.00	4.23

Table 5: Sink and recent window sizes on Qwen3-4B-Thinking-2507 (INT2 with OSCAR rotation and calibration-derived clip). (S, R) denote the number of BF16 sink tokens and recent-window tokens, respectively. "Extra BF16 KV" reports the protected BF16-token fraction, $(S + R)/128\text{K}$.

(S, R)	GPQA	HumanE	LCB v6	AIME25	MATH500	Mean	Extra BF16 KV
$(0, 0)$	0.00	0.00	0.00	0.00	0.00	0.00	0 %
$(32, 128)$	57.58±2.20	91.91±0.19	40.74±2.36	55.56±5.09	92.65±0.95	67.69	0.12 %
$(64, 256)$	**64.95±1.16**	**92.24±1.02**	**45.38±1.97**	**64.00±3.65**	**92.75±0.39**	**71.86**	**0.24 %**
$(128, 512)$	65.32±0.77	93.17±0.32	45.22±0.33	67.78±1.92	93.32±0.12	72.96	0.49 %
$(256, 1024)$	64.98±1.27	93.13±0.81	46.20±2.03	67.78±1.92	93.32±0.12	73.08	0.98 %

Clip Threshold. The clip thresholds are estimated per layer from rotated calibration activations rather than tuned on downstream tasks. Table 6 sweeps global clip ratios on Qwen3-4B-Thinking and shows that the calibration-derived point $(c_K, c_V) = (0.96, 0.92)$ is within 0.15 points of the best downstream grid point, confirming that OSCAR's clip thresholds do not need to be hand-tuned.

Calibration Data Regime. We also test whether OSCAR is sensitive to calibration volume or domain. Table 7 shows that the default 8k GPQA-Diamond calibration is close to the best setting, supporting a small one-time calibration pass.

Table 6: Validation of calibration-derived clip thresholds (mean accuracy on Qwen3-4B-Thinking-2507). Rows: c_K; columns: c_V; bold: operating point $(0.96, 0.92)$.

$c_K \setminus c_V$	0.88	0.92	0.96	0.98	1.00
0.88	67.20	68.27	68.10	67.56	68.25
0.92	69.15	70.00	70.46	70.74	70.33
0.96	68.77	**70.59**	70.70	68.17	69.14
0.98	65.22	65.76	64.12	64.79	67.31
1.00	60.35	59.19	58.15	58.63	56.02

Table 7: Calibration-data regime on Qwen3-4B-Thinking-2507 (INT2 with OSCAR rotation, calibration-derived clip, sink/recent protection), sweeping volume and domain.

Domain	Tokens	GPQA	HumanE	LCB v6	AIME25	MATH500	Mean
MMLU	2k	60.61±2.20	92.52±0.14	39.57±2.36	55.56±5.09	93.12±0.46	68.28
	8k	60.44±2.04	92.07±0.76	40.74±0.89	60.00±5.77	92.52±0.58	69.15
	16k	64.14±0.51	92.93±0.53	43.86±2.55	61.11±10.72	92.38±0.53	70.88
	32k	61.28±1.54	92.20±0.32	46.00±2.36	57.78±5.09	92.25±0.31	69.90
WikiText	8k	61.11±0.51	92.72±0.63	42.11±2.11	58.89±3.85	92.65±0.50	69.50
GPQA	**8k (def.)**	**62.96±1.27**	**92.56±0.00**	**44.64±1.47**	**62.22±1.92**	**92.65±0.31**	**71.01**

5.4 End-to-End Serving Throughput

Kernel-Level Profiling. We profile per-step decode latency on Qwen3-8B and GLM-4.7-FP8 to isolate the runtime cost of OSCAR. Table 8 shows that OSCAR reduces attention time by shrinking KV traffic while adding only a small fused quantization cost.

Table 8: Per-decode kernel profiling on Qwen3-8B, 1×H100, single decoding step latency in milliseconds.

B	BF16					OSCAR				
	GEMM	**Attn**	**Quant**	**Other**	**Total**	**GEMM**	**Attn**	**Quant**	**Other**	**Total**
1	5.6 (56.0%)	3.8 (37.9%)	–	0.6 (6.1%)	9.9	5.6 (69.9%)	1.3 (16.8%)	0.5 (5.9%)	0.6 (7.5%)	8.0
8	5.6 (53.0%)	4.4 (41.0%)	–	0.6 (6.0%)	10.7	5.7 (57.4%)	3.0 (30.1%)	0.5 (4.9%)	0.8 (7.7%)	9.9
16	5.7 (37.5%)	8.9 (58.1%)	–	0.7 (4.4%)	15.2	5.7 (49.7%)	4.6 (39.4%)	0.5 (4.2%)	0.8 (6.7%)	11.6
32	5.7 (24.3%)	17.0 (72.7%)	–	0.7 (3.0%)	23.4	5.7 (37.0%)	8.5 (54.6%)	0.5 (3.2%)	0.8 (5.2%)	15.5
64	capacity limited					5.8 (24.9%)	16.2 (69.1%)	0.5 (2.3%)	0.9 (3.7%)	23.4
128	capacity limited					5.9 (14.9%)	31.9 (81.1%)	0.6 (1.4%)	1.0 (2.6%)	39.3

Representative Serving Throughput. Table 9 reports throughput under 32 concurrent requests with 8192-token inputs and 1024-token outputs, giving the per-user and per-GPU throughput used to contextualize the speedup trends below.

Table 9: Representative end-to-end serving throughput under 32 concurrent requests (8192-token inputs, 1024-token outputs, 1×H100). U = tokens/s/user, G = tokens/s/GPU, and Acc. is the five-task mean from Table 2.

	Qwen3-4B-Thinking-2507			**Qwen3-8B**		
Method	U	G	Acc.	U	G	Acc.
BF16	41.1	1187.9	75.64	35.8	999.9	70.84
Naive INT2	66.4	1797.4	0.00	54.2	1415.1	0.00
QuaRot-INT2 [9]	66.3	1795.8	1.40	54.1	1411.9	10.14
Saw-INT4 [12]	50.6	1444.1	73.11	43.0	1183.2	69.97
OSCAR (ours)	63.3	1723.9	71.86	52.5	1374.0	69.63

Pure Decoding Speed. We first study how effective our kernel is in pure decoding speed. We isolated the prefill time or time to first token (TTFT) by setting a full prefix-cache-hit setting with a batch size 1, where all methods fully reuse the cached KV states throughout decoding. In this regime, throughput differences directly reflect decode efficiency rather than cache residency or scheduling effects. Figure 4 (Left) shows decode throughput normalized to BF16 across three models under varying context lengths. It shows that **(i)** OSCAR consistently outperforms Saw-INT4 [12] by up to 2x when the sequence length is 100k, reaching the theoretical bound, verifying the efficiency of our kernel design; **(ii)** The speedup of OSCAR increases with context length, from $1.98\times$ at 30k, $2.52\times$ at 60k, to $3.08\times$ at 100k on Qwen3-4B. This trend scales up to 358B. As such, decoding speed becomes increasingly KV-bandwidth-bound at long contexts. By reducing KV-cache memory by a factor of 8 (BF16 to INT2), OSCAR achieves substantially larger gains as the context length grows, while the online rotation overhead remains effectively hidden within the decode kernels.

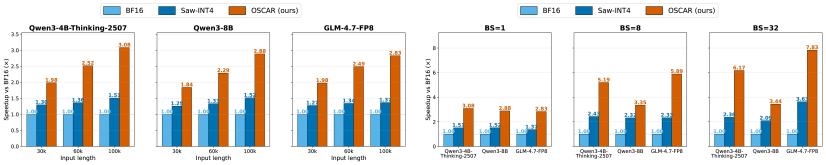

Figure 4: Left: Decode throughput speedup relative to BF16 at batch size 1 with a full prefix-cache hit and 1K output sequence length. OSCAR achieves progressively larger speedups as the context increases. Right: Job-level throughput at 100k sequence, as a function of batch size for Qwen3-4B/8B and GLM-4.7-FP8. Each group of bars compares BF16, Saw-INT4, and ours at BS $\in \{1, 8, 32\}$.

Effect of Batch Size. We next study how serving throughput scales with batch size under long-context decoding. Figure 4 (right) reports job-level throughput with prefix warm up at a fixed input length (100k tokens) with batch sizes BS $\in \{1, 8, 32\}$. Prefix warm up is used to simulate popular agentic serving workload with long prefix reuse. Three observations stand out. **(i)** OSCAR consistently outperforms BF16 across all batch sizes. On GLM-4.7-FP8, OSCAR achieves a $2.83\times$ speedup over BF16 at BS=1, which further increases to $7.83\times$ at BS=32. This behavior highlights that the reduced KV-cache footprint of INT2 methods becomes increasingly beneficial under highly concurrent long-context serving workloads. **(ii)** OSCAR consistently outperforms Saw-INT4 across all models and batch sizes in Figure 4, showing that the fused online rotation does not introduce an observable throughput penalty. **(iii)** Throughput improves substantially with larger batch sizes as

OSCAR can accommodate more concurrent requests with less memory pressure. OSCAR's design also benefits prefix caching [24], improving cache hit rates and job-level cache reuse.

Serving Throughput with Prefix Cache from 0 to 100%. Figure 5 evaluates end-to-end serving throughput under concurrent multi-user workloads by sweeping prefix-cache hit ratio from cache disabled to near-100% cache hits.

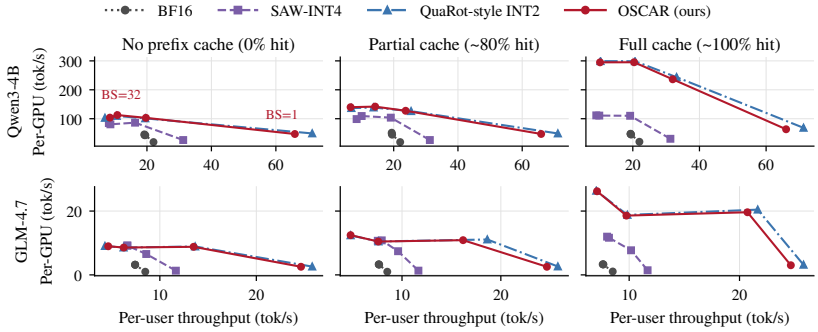

Figure 5: Effect of prefix-cache hit ratio on end-to-end serving throughput (100k ISL, 1K OSL). Each subplot plots median per-user throughput U (tok/s) against mean per-GPU throughput G (tok/s) for BS $\in \{1, 8, 16, 32\}$ on Qwen3-4B-Thinking-2507 and GLM-4.7-FP8 across 0%, $\sim$80%, and $\sim$100% prefix-cache hit regimes.

The results show that **(i)** increasing the prefix-cache hit ratio consistently expands the throughput frontier by reducing prefill recomputation, with the largest gains appearing at higher batch sizes; **(ii)** OSCAR remains on or near the efficiency frontier across all cache regimes, closely matching aggressive INT2 baselines while avoiding extra decode-time memory traffic or indirection; **(iii)** compared with BF16, low-bit KV quantization enables substantially higher per-GPU throughput because it improves decode efficiency and reduces KV-cache memory footprint, allowing larger concurrent batches under the same memory budget; and **(iv)** OSCAR achieves the best accuracy among INT2 methods while preserving the standard paged KV-cache abstraction and integrating cleanly into fused SGLang decode kernels, whereas QuaRot-INT2 can approach the throughput ceiling but suffers significant accuracy degradation as shown in Table 2.

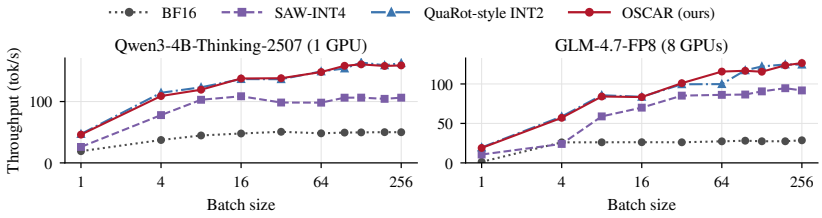

Figure 6: Long-context serving stress test with 100k-token inputs. OSCAR's uniform INT2 KV-cache scales up to 2^8 requests with continued throughput gains.

Memory and Extreme Batch Scalability. Finally, Figure 6 evaluates memory usage and batch scalability under 100k-token inputs up to 2^8 concurrent requests. By reducing the KV-cache footprint by nearly $8\times$ relative to BF16 while keeping uniform INT2 pages, OSCAR scales to 2^8 requests on a single H100 (80 GB) with continued throughput gains, whereas BF16 and INT4 baselines run out of memory or plateau early. Detailed ablations and discussion are in Appendices E.3 and D.

6 Conclusion

We introduced **OSCAR**, an INT2 KV-cache quantization method that chooses rotations from attention-aware target covariance rather than raw cache reconstruction. With one offline calibration pass, OSCAR fixes key/value rotations and clip thresholds, then uses a fused SGLang path to quantize history KV tokens while preserving sink and recent tokens. This simple target-and-kernel co-design recovers much of the BF16 accuracy lost by naive INT2 while keeping the memory and serving advantages of 2-bit KV caches.

Acknowledgments and Disclosure of Funding

We thank the anonymous reviewers and the area chair for their detailed feedback, which led to the matched-baseline comparisons, the residual diagnostics, and the long-context analysis added in this version. This work was supported by Together AI, which also provided the compute used for all experiments. Zhongzhu Zhou, Donglin Zhuang, Jisen Li, Shuaiwen Leon Song, Ben Athiwaratkun, and Xiaoxia Wu are affiliated with Together AI, which develops and operates LLM serving infrastructure; the authors declare no other competing interests.

References

[1] 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. *Advances in Neural Information Processing Systems*, 2023.

[2] Akshat Sharma, Hangliang Ding, Jianping Li, Neel Dani, and Minjia Zhang. MiniKV: Pushing the limits of LLM inference via 2-bit layer-discriminative KV cache. *arXiv preprint arXiv:2411.18077*, 2024.

[3] Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive KV cache compression for LLMs. In *International Conference on Learning Representations*, 2024.

[4] 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*, 2024.

[5] Yuxuan Yue, Zhihang Yuan, Haojie Duanmu, Sifan Zhou, Jianlong Wu, and Liqiang Nie. WKVQuant: Quantizing weight and key/value cache for large language models gains more. *arXiv preprint arXiv:2402.12065*, 2024.

[6] Haojun Xia, Xiaoxia Wu, Jisen Li, Robert Wu, Junxiong Wang, Jue Wang, Chenxi Li, Aman Singhal, Alay Dilipbhai Shah, Alpay Ariyak, Donglin Zhuang, Zhongzhu Zhou, Ben Athiwaratkun, Zhen Zheng, and Shuaiwen Leon Song. Kitty: Accurate and efficient 2-bit KV cache quantization with dynamic channel-wise precision boost. *arXiv preprint arXiv:2511.18643*, 2025.

[7] Zunhai Su, Zhe Chen, Wang Shen, Hanyu Wei, Linge Li, Huangqi Yu, and Kehong Yuan. RotateKV: Accurate and robust 2-bit KV cache quantization for LLMs via outlier-aware adaptive rotations. *arXiv preprint arXiv:2501.16383*, 2025.

[8] Haoyang LI, Yiming Li, Anxin Tian, Tianhao Tang, Zhanchao Xu, Xuejia Chen, Nicole HU, Wei Dong, Li Qing, and Lei Chen. A survey on large language model acceleration based on KV cache management. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=z3JZzu9EA3.

[9] 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*, volume 37, pages 100213–100240, 2024.

[10] 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*, volume 235 of *Proceedings of Machine Learning Research*, pages 48630–48656, 2024.

[11] 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*, 2025.

[12] Jinda Jia, Jisen Li, Zhongzhu Zhou, Jung Hwan Heo, Jue Wang, Tri Dao, Shuaiwen Leon Song, Ben Athiwaratkun, Chenfeng Xu, Tianyi Zhang, et al. SAW-INT4: System-aware 4-bit KV-cache quantization for real-world LLM serving. *arXiv preprint arXiv:2604.19157*, 2026.

[13] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph 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*, 2023.

[14] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[15] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023. URL https://arxiv.org/abs/2307.08691.

[16] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. FlashAttention-3: Fast and accurate attention with asynchrony and low-precision. In *Advances in Neural Information Processing Systems*, volume 37, pages 68658–68685, 2024.

[17] Ted Zadouri, Markus Hoehnerbach, Jay Shah, Timmy Liu, Vijay Thakkar, and Tri Dao. FlashAttention-4: Algorithm and kernel pipelining co-design for asymmetric hardware scaling. *arXiv preprint arXiv:2603.05451*, 2026.

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

[19] Federico Barbero, Álvaro Arroyo, Xiangming Gu, Christos Perivolaropoulos, Michael Bronstein, Petar Veličković, and Razvan Pascanu. Why do LLMs attend to the first token? *arXiv preprint arXiv:2504.02732*, 2025.

[20] 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*, volume 235 of *Proceedings of Machine Learning Research*, 2024.

[21] Haojie Duanmu, Zhihang Yuan, Xiuhong Li, Jiangfei Duan, Xingcheng Zhang, and Dahua Lin. SKVQ: Sliding-window key and value cache quantization for large language models. In *Conference on Language Modeling*, 2024.

[22] Zhongzhu Zhou and Donglin Zhuang. [feature] OSCAR mixed-precision INT2 KV cache (–kv-cache-dtype int2). https://github.com/sgl-project/sglang/pull/32129, 2026. SGLang pull request #32129.

[23] Jian Zhang. A new 2-bit KV cache quantisation backend (OSCAR). https://github.com/vllm-project/vllm/pull/46774, 2026. vLLM pull request #46774, under review.

[24] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, 2024.

[25] 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*, 2017.

[26] 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*, pages 4895–4901, 2023.

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

[28] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301, 1965. doi: 10.1090/S0025-5718-1965-0178586-1.

[29] Charles Van Loan. *Computational Frameworks for the Fast Fourier Transform*, volume 10 of *Frontiers in Applied Mathematics*. SIAM, Philadelphia, PA, 1992.

[30] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[31] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations*, 2023.

[32] 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*, 2019.

[33] Tri Dao, Daniel Haziza, Francisco Massa, and Grigory Sizov. Flash-decoding for long-context inference. https://pytorch.org/blog/flash-decoding/, 2023. PyTorch Blog. Accessed: 2026-05-06.

[34] Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[35] GLM Team. ChatGLM: A family of large language models from GLM-130B to GLM-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024.

[36] Mathematical Association of America. AIME 2025: American invitational mathematics examination. https://maa.org/math-competitions/aime, 2025.

[37] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof Q&A benchmark. In *First Conference on Language Modeling*, 2024.

[38] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[39] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, 2025.

[40] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

[41] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024.

[42] 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 *First Conference on Language Modeling*, 2024.

[43] vibhavagarwal5. TurboQuant: 2-bit KV cache compression with 4x capacity. https://github.com/vllm-project/vllm/pull/38479, 2026. vLLM pull request #38479.

[44] Amir Zandieh, Majid Daliri, Majid Hadian, and Vahab Mirrokni. TurboQuant: Online vector quantization with near-optimal distortion rate. *arXiv preprint arXiv:2504.19874*, 2025.

[45] Sanae Lotfi, Polina Kirichenko, Steven Li, and Zechun Liu. Quantized reasoning models think they need to think longer, but they do not. *arXiv preprint arXiv:2606.00206*, 2026.

[46] Ky Fan. Maximum properties and inequalities for the eigenvalues of completely continuous operators. *Proceedings of the National Academy of Sciences of the United States of America*, 37 (11):760–766, 1951. doi: 10.1073/pnas.37.11.760.

[47] 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*, 2024.

[48] Zefan Cai et al. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

[49] 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 *Advances in Neural Information Processing Systems*, 2024.

[50] Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. GEAR: An efficient KV cache compression recipe for near-lossless generative inference of LLM. *arXiv preprint arXiv:2403.05527*, 2024.

[51] 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: Compressing KV-cache with low-rank projection. In *International Conference on Learning Representations*, 2025.

[52] Chi-Chih Chang, Chien-Yu Lin, Yash Akhauri, Wei-Cheng Lin, Kai-Chiang Wu, Luis Ceze, and Mohamed S. Abdelfattah. xKV: Cross-layer SVD for KV-cache compression. *arXiv preprint arXiv:2503.18893*, 2025.

[53] Bokai Lin, Zihao Zeng, Zipeng Xiao, Siqi Kou, Tianqi Hou, Xiaofeng Gao, Hao Zhang, and Zhijie Deng. MatryoshkaKV: Adaptive KV compression via trainable orthogonal projection. *arXiv preprint arXiv:2410.14731*, 2024.

[54] 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-CoT LLMs. *arXiv preprint arXiv:2505.18610*, 2025.

[55] Mohsen Hariri, Alan Luo, Weicong Chen, Shaochen Zhong, Tianyi Zhang, Qifan Wang, Xia Hu, Xiaotian Han, and Vipin Chaudhary. Quantize what counts: More for keys, less for values. *arXiv preprint arXiv:2502.15075*, 2025.

[56] Saleh Ashkboos, Mahdi Nikdan, Soroush Tabesh, Roberto L. Castro, Torsten Hoefler, and Dan Alistarh. HALO: Hadamard-assisted lower-precision optimization for LLMs. *arXiv preprint arXiv:2501.02625*, 2025.

[57] Seonggon Kim, Juncheol Shin, Seung-taek Woo, and Eunhyeok Park. HOT: Hadamard-based optimized training. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 4787–4796, 2025.

[58] Vage Egiazarian, Roberto L. Castro, Denis Kuznedelev, Andrei Panferov, Eldar Kurtic, Shubhra Pandit, Alexandre Noll Marques, Mark Kurtz, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Bridging the gap between promise and performance for microscaling FP4 quantization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=zCBGe9AqJZ.

[59] Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. DuQuant: Distributing outliers via dual transformation makes stronger quantized LLMs. In *Advances in Neural Information Processing Systems*, volume 37, pages 87766–87800, 2024.

[60] Sai Sanjeet, Ian Colbert, Pablo Monteagudo-Lago, Giuseppe Franco, Yaman Umuroglu, and Nicholas J. Fraser. MixQuant: Pushing the limits of block rotations in post-training quantization. *arXiv preprint arXiv:2601.22347*, 2026.

[61] Utkarsh Saxena and Kaushik Roy. KVLinC: KV cache quantization with Hadamard rotation and linear correction. *arXiv preprint arXiv:2510.05373*, 2025.

[62] Jiale Chen, Vage Egiazarian, Roberto L. Castro, Torsten Hoefler, and Dan Alistarh. WUSH: Near-optimal adaptive transforms for LLM quantization. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 14118–14144, 2026.

[63] Patrick H. Chen, Hsiang-Fu Yu, Inderjit S. Dhillon, and Cho-Jui Hsieh. DRONE: Data-aware low-rank compression for large NLP models. *Advances in Neural Information Processing Systems*, 34:29321–29334, 2021.

[64] Zhihang Yuan, Yuzhang Shang, Yang Song, Qiang Wu, Yan Yan, and Guangyu Sun. ASVD: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv:2312.05821*, 2023.

[65] Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. SVD-LLM: Truncation-aware singular value decomposition for large language model compression. In *International Conference on Learning Representations*, 2025.

[66] Xin Wang, Samiul Alam, Zhongwei Wan, Hui Shen, and Mi Zhang. SVD-LLM v2: Optimizing singular value truncation for large language model compression. In *Proceedings of the Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2025.

[67] Yibo Yang, Xiaojie Li, Zhongzhu Zhou, Shuaiwen Leon Song, Jianlong Wu, Liqiang Nie, and Bernard Ghanem. CorDA: Context-oriented decomposition adaptation of large language models. *arXiv preprint arXiv:2406.05223*, 2024.

[68] Seonggon Kim, Taehyeon Kim, and Eunhyeok Park. HaPPI: Efficient KV cache compression with Hadamard PCA-based power iteration. OpenReview, 2025. URL https://openreview.net/forum?id=BRDgQzdtWr. Submitted to ICLR 2026.

[69] Zhongzhu Zhou, Fengxiang Bie, Ziyan Chen, Zhenyu Zhang, Yibo Yang, Junxiong Wang, Ben Athiwaratkun, Xiaoxia Wu, and Shuaiwen Leon Song. CARE: Covariance-aware and rank-enhanced decomposition for enabling multi-head latent attention. In *International Conference on Learning Representations*, 2026.

[70] Xianglong Yan, Zhiteng Li, Tianao Zhang, 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.

[71] Yixuan Wang, Haoyu Qiao, Lujun Li, Qingfu Zhu, and Wanxiang Che. CommonKV: Compressing KV cache with cross-layer parameter sharing. *arXiv preprint arXiv:2508.16134*, 2025.

[72] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, 2024.

[73] Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. OmniQuant: Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations*, 2024.

[74] Akshat Ramachandran, Marina Neseem, Charbel Sakr, Rangharajan Venkatesan, Brucek Khailany, and Tushar Krishna. ThinKV: Thought-adaptive KV cache compression for efficient reasoning models. *arXiv preprint arXiv:2510.01290*, 2025. URL https://arxiv.org/abs/2510.01290.

[75] Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The Berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

[76] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

A Additional Details and Theoretical Analysis

A.1 Hadamard Transform

The Hadamard transform is an orthogonal mixing transform with entries of equal magnitude. We use the normalized Walsh-Hadamard matrix H_{Had}, which satisfies $H_{\mathrm{Had}}^{\top} H_{\mathrm{Had}} = I$. Its dimension is inferred from the key or value vector being transformed. For power-of-two dimensions, it can be defined recursively by $H_1 = [1]$ and

$$H_{2m} = \frac{1}{\sqrt{2}} \begin{bmatrix} H_m & H_m \\ H_m & -H_m \end{bmatrix}.$$

In low-bit LLM quantization, Hadamard transforms are commonly used to spread outlier energy across channels before rounding [9]. In OSCAR, H_{Had} is composed after the attention-aware covariance rotation to further smooth the rotated coordinates while keeping the transform fixed and efficient.

A.2 Principal Component Analysis

Given a symmetric positive semidefinite matrix $A \in \mathbb{R}^{d\times d}$, principal component analysis (PCA) diagonalizes it via

$$A = V\Lambda V^{\top}, \qquad \Lambda = \mathrm{diag}(\lambda_1, \dots, \lambda_r, \dots), \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq 0.$$

Below we recall the important proposition of PCA [46].

Proposition 1 (Indication of top-r eigenvectors)**.** *For any $1 \leq r \leq d$, let*

$$V_r = [v_1, \dots, v_r] \in \mathbb{R}^{d\times r}.$$

Then

$$V_r = \mathrm{argmax}_{U^{\top}U = I_r} \mathrm{tr}(U^{\top} A U),$$

and the optimal value is

$$\max_{U^{\top}U = I_r} \mathrm{tr}(U^{\top} A U) = \sum_{i=1}^{r} \lambda_i.$$

Note that in our setting, when A is chosen as a target covariance induced by the attention computation, PCA provides an orthogonal basis aligned with the directions that are most important to preserve under KV compression.

Proof. Let $U \in \mathbb{R}^{d\times r}$ satisfy $U^{\top}U = I_r$. Since $A = V\Lambda V^{\top}$, we have

$$\mathrm{tr}(U^{\top} A U) = \mathrm{tr}(U^{\top} V \Lambda V^{\top} U).$$

Define

$$B = V^{\top} U \in \mathbb{R}^{d\times r}.$$

Because V is orthogonal,

$$B^{\top} B = U^{\top} V V^{\top} U = U^{\top} U = I_r.$$

Hence the columns of B are orthonormal. Using cyclicity of trace,

$$\mathrm{tr}(U^{\top} A U) = \mathrm{tr}(B^{\top} \Lambda B) = \mathrm{tr}(\Lambda B B^{\top}).$$

Write

$$P = BB^{\top} \in \mathbb{R}^{d\times d}.$$

Then P is an orthogonal projection matrix of rank r, so

$$0 \leq P_{ii} \leq 1, \qquad \sum_{i=1}^{d} P_{ii} = \mathrm{tr}(P) = r.$$

Therefore

$$\mathrm{tr}(U^{\top} A U) = \mathrm{tr}(\Lambda P) = \sum_{i=1}^{d} \lambda_i P_{ii}.$$

Set

$$w_i = P_{ii}, \qquad i = 1, \dots, d.$$

Then

$$0 \le w_i \le 1, \qquad \sum_{i=1}^{d} w_i = r,$$

and thus

$$\operatorname{tr}(U^\top A U) = \sum_{i=1}^{d} \lambda_i w_i.$$

Since $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_d$, the quantity $\sum_{i=1}^{d} \lambda_i w_i$ is maximized by assigning full weight to the largest r eigenvalues, namely

$$w_1 = \cdots = w_r = 1, \qquad w_{r+1} = \cdots = w_d = 0.$$

Hence

$$\operatorname{tr}(U^\top A U) \le \sum_{i=1}^{r} \lambda_i.$$

Now choose $U = V_r$. Then

$$V_r^\top A V_r = V_r^\top V \Lambda V^\top V_r = \operatorname{diag}(\lambda_1, \dots, \lambda_r),$$

so

$$\operatorname{tr}(V_r^\top A V_r) = \sum_{i=1}^{r} \lambda_i.$$

Thus V_r attains the maximum, and therefore

$$V_r \in \operatorname{argmax}_{U^\top U = I_r} \operatorname{tr}(U^\top A U).$$

□

A.3 Target Covariance

Given a matrix $X = [x_1; \dots; x_N] \in \mathbb{R}^{N \times d}$ whose rows are feature vectors, its empirical covariance, assuming the rows have been centered, is

$$C_X = \frac{1}{N} X^\top X = \frac{1}{N} \sum_{i=1}^{N} x_i^\top x_i \in \mathbb{R}^{d \times d}.$$

In this paper, we use the uncentered second moment in the same algebraic form and refer to it as a covariance target. For any direction $u \in \mathbb{R}^d$ with $\|u\|_2 = 1$, the scalar

$$u^\top C_X u = \frac{1}{N} \sum_{i=1}^{N} (x_i u)^2$$

measures the average squared magnitude of the rows of X along direction u. Thus, the eigenvectors of C_X identify directions where the row activations have large energy, and PCA uses these eigenvectors as a basis aligned with the dominant second-order structure of X.

In OSCAR, the covariance target is not chosen only from the raw cached tensors. The reason is that the rotation should reduce the downstream attention error, not merely the Euclidean reconstruction error of K and V. For keys, the empirical logit distortion satisfies

$$\|QK^\top - Q\widehat{K}^\top\|_F^2 = \operatorname{tr}\Big((K - \widehat{K}) Q^\top Q (K - \widehat{K})^\top\Big).$$

This identity shows that key errors are weighted by the query covariance $Q^\top Q$. Heuristically, directions with large eigenvalues of $Q^\top Q$ are directions in which the queries have large energy, so quantization error in those directions causes larger perturbations to the attention logits. Therefore, we use $Q^\top Q$ as the key-side target covariance for determining the raw key rotation.

For values, the downstream output distortion satisfies

$$\|SV - S\widehat{V}\|_F^2 = \mathrm{tr}\Big((V - \widehat{V})^\top S^\top S(V - \widehat{V})\Big).$$

Equivalently, the matrix

$$V^\top S^\top SV = (SV)^\top (SV)$$

is the target covariance of the attention-weighted value activations SV. It measures which value feature directions remain large after aggregation by the attention scores. Hence, $V^\top S^\top SV$ is an attention-induced value target covariance, while $V^\top V$ is only the raw-cache covariance of values.

Therefore, $Q^\top Q$ and $V^\top S^\top SV$ are covariance targets in our context: they are second-moment matrices induced by the downstream attention computation. This is why OSCAR estimates rotations from these attention-aware covariance targets rather than from raw-cache targets such as $K^\top K$ and $V^\top V$.

A.4 Intuition of the combination $R_K = U_Q H_{\mathrm{Had}} P_K$

The rotation applied to keys before quantization in OSCAR is the product of three orthogonal factors,

$$R_K \;=\; U_Q \, H_{\mathrm{Had}} \, P_K,$$

where U_Q is the eigenbasis of the key target covariance $C_Q = Q^\top Q$ (Sec. A.3), H_{Had} is the normalized Walsh–Hadamard matrix (Sec. A.1), and P_K is a permutation matrix that we describe below. The same construction is used for R_V with $C_S = V^\top S^\top SV$ in place of C_Q. This subsection explains, before proceeding to the formal proof in Sec. A.6, why each factor is needed and why they appear in this order. Each factor addresses a distinct failure mode of per-group low-bit quantization.

Two effects of R_K. Recall from Sec. A.3 that the rotated cache $\tilde{K} = KR_K$ is what the quantizer sees, and the resulting attention-logit distortion is

$$\left\|QK^\top - Q\widehat{K}^\top\right\|_F^2 \;=\; \mathrm{tr}\big(R_K^\top C_Q R_K \cdot E_K\big),$$

where E_K is the residual covariance of $\tilde{K} - \mathcal{Q}(\tilde{K})$. The rotation enters in two distinct places:

(i) it shapes $R_K^\top C_Q R_K$, the query-importance metric in the rotated coordinates;

(ii) it shapes E_K, because the quantizer's per-token, per-group min–max scale (Sec. A.5) depends on the channelwise distribution of $\tilde{K}$.

The three factors U_Q, H_{Had}, P_K each address one of these effects in turn.

U_Q: align channels with query-importance directions. Eigendecompose

$$C_Q \;=\; U_Q \Lambda_Q U_Q^\top, \qquad \Lambda_Q = \mathrm{diag}(\lambda_1, \ldots, \lambda_d), \qquad \lambda_1 \geq \cdots \geq \lambda_d \geq 0.$$

Setting $R_K = U_Q$ diagonalizes the importance metric:

$$R_K^\top C_Q R_K \;=\; \Lambda_Q.$$

Geometrically, the j-th channel of $\tilde{K} = KU_Q$ is the projection of each key onto the j-th query principal direction u_j, and $\lambda_j = \|Qu_j\|_2^2$ measures how much total query energy lies along u_j. Quantization noise on channel j then propagates to the attention logits with weight λ_j: large-λ channels are exactly those whose error is most amplified, while small-λ channels can absorb error almost for free. Sec. A.6 formalizes this via a rearrangement-inequality argument on the surrogate $\tilde{\mathcal{L}}_K(R_k) = \mathrm{tr}(R_k^\top C_Q R_k E_K)$: under the frozen-residual assumption, $R_K = U_Q$ is the orthogonal rotation that minimizes the importance-weighted error.

This is the right first step but it leaves a per-group quantization problem. The eigenspectrum Λ_Q of C_Q in real LLM activations is sharply anisotropic: a few eigenvalues carry most of $\mathrm{tr}(C_Q)$. After $R_K = U_Q$, the rotated key channels inherit this imbalance, with a few channels having much larger variance than the rest. A per-group min–max quantizer with a single scale per group then sets its scale based on the largest channel in each group, and the smaller channels in the same group are quantized at a granularity far coarser than their dynamic range. Hence E_K has a few very large diagonal entries and many tiny ones, and $\mathrm{tr}(E_K)$ is dominated by the worst groups.

H_{Had}: equalize on the importance metric, suppress outliers on K. The U_Q step is chosen for the surrogate, which is a Q-side criterion ($C_Q = Q^\top Q$). The Hadamard composition serves two complementary purposes: on the surrogate side, it equalizes the rotated importance metric across channels; on the actual K distribution, it mixes outlier energy across channels so that the per-group min–max quantizer (Sec. A.5) sees comparable per-channel magnitudes. The first claim is exact; the second is what the standard outlier-suppression literature relies on [9] and only requires that the Hadamard column weights all have equal magnitude.

The exact statement on the importance metric is the following.

Lemma 1 (Diagonal equalization under Hadamard conjugation)**.** *Let $\Lambda \in \mathbb{R}^{d\times d}$ be diagonal and H_{Had} the normalized Walsh–Hadamard matrix. Then*

$$\left(H_{\text{Had}}^\top \Lambda\, H_{\text{Had}}\right)_{ii} \;=\; \frac{1}{d}\text{tr}(\Lambda), \qquad i = 1, \ldots, d.$$

Proof. Every entry of H_{Had} has magnitude $1/\sqrt{d}$. Then $(H_{\text{Had}}^\top \Lambda H_{\text{Had}})_{ii} = \sum_j H_{\text{Had}}[j,i]^2\, \Lambda_{jj} = \frac{1}{d}\sum_j \Lambda_{jj} = \frac{1}{d}\text{tr}(\Lambda)$, independent of i. □

Applying the lemma to $\Lambda = \Lambda_Q$ (which is diagonal because of the U_Q step) gives

$$\left(R_K^\top C_Q R_K\right)_{ii} \;=\; \left(H_{\text{Had}}^\top \Lambda_Q H_{\text{Had}}\right)_{ii} \;=\; \frac{1}{d}\text{tr}(C_Q), \qquad \forall\, i.$$

The peaky eigenspectrum that U_Q exposed has been replaced by a uniform diagonal: every channel now carries the same importance weight $\text{tr}(C_Q)/d$ in the rotated frame.

The K-side effect is heuristic but practically important. After $R_K = U_Q H_{\text{Had}}$, each row of $\tilde{K} = KR_K$ is a Walsh-signed sum of the principal-direction projections KU_Q, with every entry of weight $\pm 1/\sqrt{d}$. A per-token outlier on any single principal direction is thus redistributed in magnitude $1/\sqrt{d}$ across all d channels, shrinking the per-token max by roughly the same factor and bringing the per-group min–max scale into a much tighter range. The frozen residual E_K becomes much smaller in trace and far more uniform across i. The price paid is that the importance metric is no longer diagonal: $R_K^\top C_Q R_K = H_{\text{Had}}^\top \Lambda_Q H_{\text{Had}}$ has off-diagonals that couple channels. In the regime where per-group quantization noise dominates $\text{tr}(E_K)$, the reduction in $\text{tr}(E_K)$ wins over the loss of importance alignment.

Empirical aside: U_Q does *not* accidentally diagonalize Σ_K. One might hope that, because W_Q and W_K are trained on the same activation distribution, $C_Q = Q^\top Q$ and $\Sigma_K = K^\top K$ would share an eigenbasis, so that U_Q would simultaneously serve both the Q-side (importance) and K-side (variance) objectives. Empirically, this is not the case. On Qwen3-8B with 2000 calibration tokens per layer, the top-8 eigenvector self-alignment $\sum_{k=1}^{8} |U_Q^{Q\top} U_Q^K|/8$ ranges from 0.05 to 0.15 across the 36 layers — essentially indistinguishable from the random alignment $1/\sqrt{d} \approx 0.09$ expected at $d = 128$. Rotating Σ_K by U_Q typically *reduces* the fraction of Σ_K's energy concentrated on the diagonal (e.g., layer 1: 0.90 raw vs. 0.09 after U_Q rotation; layer 16: 0.16 vs. 0.07). Figure 7 shows a representative layer.

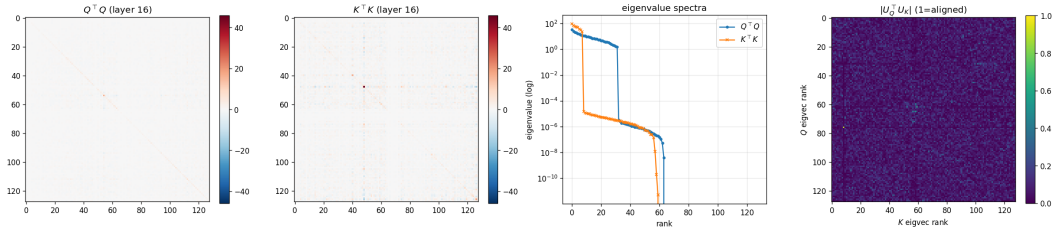

Figure 7: Layer 16 of Qwen3-8B. From left: heatmap of $Q^\top Q$; heatmap of $K^\top K$ (same colour scale); eigenvalue spectra; absolute eigenvector alignment $|U_Q^\top U_Q|$ with both bases sorted by descending eigenvalue. A diagonal-bright alignment matrix would indicate a shared eigenbasis; the observed uniform-texture pattern shows the eigenbases are nearly uncorrelated.

The construction is nevertheless robust because the two factors target different things. Lemma 1 guarantees exact diagonal equalization of the importance metric C_Q regardless of the eigenbasis of Σ_K, since the lemma only relies on Λ_Q being diagonal in the post-U_Q frame. The K-side outlier-suppression effect of H_{Had} described above also does not require alignment between U_Q and the eigenbasis of Σ_K: the Walsh-signed mixing shrinks per-token outliers by a factor of $1/\sqrt{d}$ no matter which orthogonal basis K was placed in beforehand. The Q-side (U_Q) and K-side (H_{Had}) factors thus address genuinely independent objectives and compose without interference, rather than relying on an approximate $W_Q \approx W_K$ similarity that does not hold in practice.

P_K: balance importance across groups. Hadamard mixing equalizes *marginal* channel variance, but the per-group quantizer in Sec. A.5 acts on contiguous blocks of G_K channels at a time. What matters at the group level, beyond marginal variance, is whether each group sees a balanced sample of the underlying importance directions: a group whose G_K channels are dominated by the same handful of large-λ_j eigenvectors is statistically harder to quantize than a group whose channels span the full range of λ_j.

OSCAR uses a *permuted bit-reversal* (PBR) ordering. Let σ be the descending-eigenvalue permutation, so that $\lambda_{\sigma(0)} \geq \lambda_{\sigma(1)} \geq \cdots$, and let β be the bit-reversal permutation on $\{0, 1, \ldots, d-1\}$. The permutation matrix P_K places the eigenvector with the k-th largest eigenvalue at position $\beta(k)$, i.e. $(P_K)_{:,\beta(k)} = e_{\sigma(k)}$. For $d = 128$, this yields the placement

$$\text{top-1} \mapsto 0, \quad \text{top-2} \mapsto 64, \quad \text{top-3} \mapsto 32, \quad \text{top-4} \mapsto 96, \quad \text{top-5..8} \mapsto 16, 80, 48, 112, \quad \ldots$$

The defining property of bit-reversal is that for any power-of-two group size $G \mid d$, the top-d/G eigenvectors land in d/G *distinct* groups, exactly one per group; the same recursive balance holds at every coarser binary level. Composed with the Hadamard, this means each per-group quantization block sees one representative from each level of the importance hierarchy, regardless of which group size G_K is chosen at deployment.

Empirical aside: errors separate after attention. Figure 8 compares layer-wise errors on Qwen3-4B-Thinking-2507, measured on raw KV tensors and on the downstream quantities consumed by attention. OSCAR is not designed to minimize plain Euclidean reconstruction error of K or V, so its raw K-MSE and V-MSE are not always dramatically smaller than rotation-only baselines. The gap becomes much clearer after the attention computation: the errors in $QK^\top$ and SV are substantially lower, and the advantage further propagates to attention-output errors. This is exactly what the covariance targets optimize: the key rotation is aligned with query-side covariance, while the value rotation is aligned with the score-weighted value covariance. The trend is consistent with the U_K intuition above: aligning channels with query-importance directions need not minimize raw K error, but it directly lowers the error after multiplication by Q; analogously, the value target is chosen so that the advantage appears after aggregation by S.

Worked example: Qwen3-4B-Thinking, layer 10, $d = 128$. We illustrate the three factors on real activations rather than a stylized spectrum. Calibration sample: layer 10 of Qwen3-4B-Thinking-2507 (32 query heads, 8 key/value heads, GQA ratio $= 4$), $T = 8000$ tokens of GPQA-diamond context. We use the production importance metric — Q-side query covariance, averaged within each GQA group of 4 query heads and then across all 8 kv-heads,

$$C_Q \;=\; \frac{1}{H_{\text{kv}}} \sum_{h=1}^{H_{\text{kv}}} \frac{1}{T \cdot g} \sum_{i \in \mathcal{G}_h} Q_i^\top Q_i,$$

The decode-time per-group min–max quantizer uses INT2 with $G_K = 64$ (two groups per head); we deliberately pick $G_K < d$ here so the PBR step has a non-trivial effect (see Step 3 for the $G_K = d$ degenerate case). The K activations we display are from kv-head 0; the shared rotation $R_K = U_Q H_{\text{Had}} P_K$ is the one produced by `compute_qqt`.

Spectrum at layer 10. Eigendecomposing $C_Q = U_Q \Lambda_Q U_Q^\top$ yields an anisotropic but not pathologically peaky spectrum (sorted descending),

$$\lambda_{1\ldots8} \approx (162.5,\ 85.5,\ 35.6,\ 17.4,\ 11.6,\ 7.5,\ 6.1,\ 5.0),$$
$$\lambda_{121\ldots128} \approx (0.24,\ 0.22,\ 0.22,\ 0.20,\ 0.18,\ 0.05,\ 0.027,\ 0.009).$$

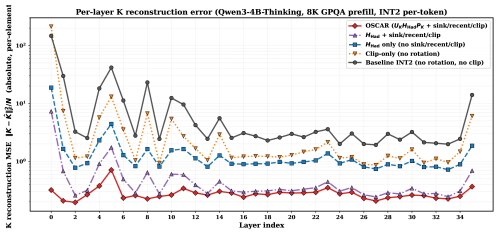

Figure 8a: K MSE

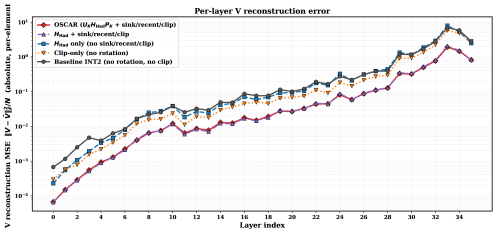

Figure 8b: V MSE

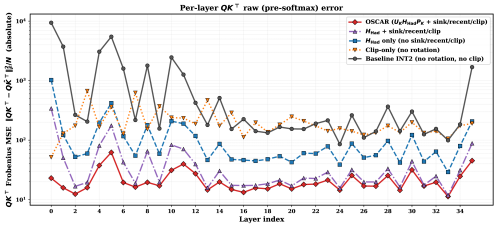

Figure 8c: $QK^\top$ MSE

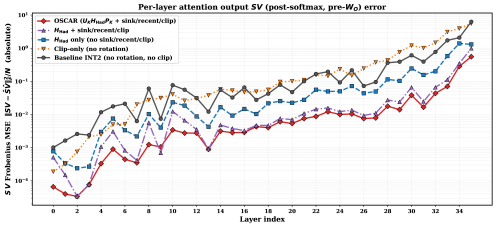

Figure 8d: SV MSE

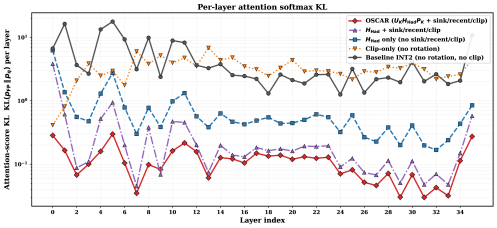

Figure 8e: Attention KL

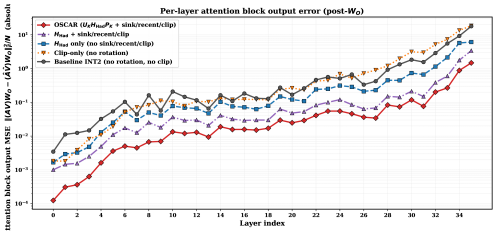

Figure 8f: Attention-output MSE

Figure 8: **OSCAR's advantage appears on attention-consumed quantities on Qwen3-4B-Thinking-2507.** Raw K/V reconstruction MSE alone does not fully explain downstream quality, because OSCAR does not optimize raw-cache reconstruction. Instead, it uses query- and score-aware covariance targets, which more directly reduce $QK^\top$, SV, and attention-output errors.

The trace is $\mathrm{tr}(C_Q) \approx 443$. The top eigendirection λ_1 alone accounts for 36.6% of the trace and the top eight carry 74.7%: $\lambda_1/\bar{\lambda} \approx 46.9$ where $\bar{\lambda} = \mathrm{tr}(C_Q)/d = 3.46$. Importance is concentrated in a low-rank subspace, but not as extremely as one might guess from K's own variance — instead, covariance $Q^\top Q$ reflects what the attention layer *actually queries*, which is a softer concentration than where K's variance happens to land.

One token, all 128 channels, split into two $G_K = 64$ groups. The INT2 per-group min–max quantizer fits its scale to $\big(\max_{i\in g} \tilde{K}_{t,i} - \min_{i\in g} \tilde{K}_{t,i}\big)/3$ per token t per group g. We visualize what the quantizer sees by printing all 128 channels for a single typical non-sink token ($t = 5$, kv-head 0), reshaped into an 8×16 grid (channel j at grid position $(\lfloor j/16 \rfloor,\ j \bmod 16)$). The horizontal rule splits the grid into the two $G_K = 64$ groups: rows 0–3 are group 0 (channels 0–63), rows 4–7 are group 1 (channels 64–127). Each matrix is annotated with the per-group max–min.

The raw token in the channel basis:

$$K_{t,\cdot} = \left[\begin{array}{cccccccccccccccc}
3.42 & -0.61 & 1.59 & 0.27 & -3.14 & -0.05 & 3.55 & 1.09 & 2.05 & -1.05 & 1.86 & -0.03 & -0.42 & -0.89 & -1.54 & 2.06 \\
-1.74 & -0.15 & -0.39 & -0.89 & -0.02 & 2.55 & 0.11 & 1.79 & -0.28 & 0.25 & -1.40 & -6.09 & -1.04 & -0.12 & 0.64 & -0.95 \\
9.06 & 1.17 & 0.08 & -1.49 & 0.29 & 1.13 & 0.47 & 0.17 & 1.22 & 0.55 & 14.19 & -0.26 & -0.09 & 0.20 & 2.39 & 0.40 \\
1.09 & -0.06 & \mathbf{-30.62} & 0.00 & 1.03 & 0.57 & -1.30 & 2.47 & -0.11 & -0.08 & 0.56 & -0.47 & 0.45 & -0.88 & 2.31 & -1.05 \\
\hline
0.50 & 0.44 & 0.10 & -0.35 & -0.78 & 1.46 & -1.16 & 2.56 & 1.64 & -0.09 & 0.90 & 2.33 & 1.09 & 2.72 & 0.95 & 0.20 \\
1.30 & 1.82 & 0.54 & 1.27 & -1.05 & 0.50 & -4.47 & 1.20 & 0.32 & 1.36 & -0.81 & -0.04 & 2.27 & -0.77 & 0.64 & 1.27 \\
0.03 & -0.20 & 0.41 & 0.58 & 0.79 & 1.05 & 0.46 & -0.09 & 0.49 & 0.29 & -0.02 & 0.20 & 1.36 & 0.77 & -1.01 & -2.33 \\
1.19 & -1.77 & -0.01 & -0.49 & 0.47 & 0.91 & 0.42 & 0.17 & 1.38 & -0.39 & -0.00 & 0.43 & -0.98 & -0.50 & 0.17 & -0.50
\end{array}\right]$$

group 0: $\max - \min = \mathbf{44.81}$ | group 1: $\max - \min = \mathbf{7.19}$.

The two groups are wildly unbalanced: group 0 holds both outliers (-30.62 at $j = 50$ and 14.19 at $j = 42$); group 1 is benign. Group 0's INT2 scale must fit a range of 44.81, wasting two bits on a few extreme values while the other ~ 60 channels of that group quantize at granularity an order of magnitude coarser than their actual dynamic range.

Step 1: $\tilde{K} = KU_Q$ (eigenbasis rotation). Projecting K onto the $Q^\top Q$ eigendirections gives:

$$(KU_Q)_{t,\cdot} = \left[\begin{array}{rrrrrrrrrrrrrrrr} 0.16 & -1.75 & 1.86 & -2.42 & 1.11 & 1.05 & -0.69 & 0.26 & 1.11 & -0.55 & 3.39 & -1.80 & 1.78 & 1.65 & -1.49 & -1.32 \\ 0.05 & -0.18 & -2.00 & -1.27 & 3.20 & -1.47 & 0.31 & -1.07 & 0.87 & -1.54 & -3.78 & -0.89 & 1.60 & 1.03 & 0.50 & 0.55 \\ 0.49 & 0.55 & -0.90 & -0.05 & 0.17 & 0.20 & 0.78 & -1.21 & -1.06 & -1.70 & 0.63 & 0.12 & -0.28 & 2.99 & 2.88 & -0.97 \\ -1.42 & 2.12 & -0.94 & -2.88 & 1.30 & -0.42 & 1.37 & 0.58 & -1.46 & -2.04 & 0.28 & -0.28 & -2.63 & -0.20 & -1.77 & -0.83 \\ \hline 0.50 & 0.12 & -1.19 & 0.81 & -1.90 & 0.40 & -1.31 & -0.15 & -2.31 & 1.52 & 3.20 & -0.46 & -0.77 & -0.09 & -1.34 & 1.72 \\ 0.40 & -0.72 & -0.57 & 0.40 & 0.84 & -0.72 & -1.94 & -0.11 & 1.26 & 0.07 & 0.09 & -0.06 & 0.87 & -0.92 & -0.63 & 1.19 \\ -2.31 & 1.56 & 1.90 & 1.26 & 1.16 & -0.72 & -0.56 & -0.48 & 1.21 & 0.20 & -0.26 & -1.78 & 1.71 & -1.28 & 1.07 & 1.27 \\ -3.32 & 0.47 & -0.43 & -1.77 & -1.42 & 1.42 & 4.99 & -1.91 & -0.25 & -1.02 & 0.51 & -1.65 & -1.11 & -8.07 & 15.28 & \mathbf{29.77} \end{array}\right]$$

group 0: $\max - \min = \mathbf{7.16}$ | group 1: $\max - \min = \mathbf{37.85}$.

The raw outlier hasn't been destroyed; it has been *redistributed* into the $Q^\top Q$ eigenbasis. Because U_Q is built from $Q^\top Q$, not $K^\top K$, the K-spike at $j = 50$ does not project onto a single U_Q direction — it spreads across the lower-eigenvalue tail with its largest projection landing in the bottom-rank eigenchannels (the matrix's bottom-right entries: $29.77, 15.28, -8.07, \ldots$). The diagonal of $U_Q^\top C_Q U_Q$ is exactly Λ_Q, so the per-channel *importance* is now exactly the eigenvalue, but the K-variance has been pushed into low-importance channels with no improvement in per-group dynamic range — in fact the imbalance between groups has flipped and grown: group 1's range jumps from 7.19 (raw) to 37.85, while group 0's drops from 44.81 to 7.16. Per-group INT2 is now harder, not easier.

Step 2: $\tilde{K} = KU_Q H_{\text{Had}}$ (Walsh-Hadamard mixing). Multiplying by the orthonormal Walsh-Hadamard $H_{\text{Had}} \in \mathbb{R}^{128\times128}$ rewrites each output channel as a ±1-weighted sum of all 128 eigenchannels (each scaled by $1/\sqrt{128}$). The single-channel spike of 29.77 is distributed across all 128 outputs with mixed signs:

$$(KU_Q H_{\text{Had}})_{t,\cdot} = \left[\begin{array}{rrrrrrrrrrrrrrrr} 2.24 & 1.08 & -3.73 & 0.26 & -5.76 & 0.92 & 4.08 & -3.39 & -3.72 & 0.29 & 5.55 & -2.67 & 3.19 & -3.70 & -4.16 & 1.80 \\ -0.87 & 0.64 & 4.12 & -3.18 & 5.12 & 1.16 & -6.79 & 0.97 & 1.91 & -2.08 & -4.91 & 4.21 & -1.67 & 1.97 & 4.47 & -0.63 \\ -2.86 & 0.66 & 6.73 & 1.83 & 4.37 & 1.49 & -5.77 & 0.72 & 1.52 & -0.05 & -4.94 & 3.97 & -3.09 & 2.28 & 4.34 & -0.37 \\ 1.90 & -1.30 & -6.06 & -1.90 & -2.58 & 3.27 & 4.18 & -3.10 & -2.31 & 0.88 & 5.04 & -2.61 & -0.07 & -1.10 & -2.17 & 4.09 \\ \hline -3.71 & 2.31 & 6.89 & -1.79 & 1.12 & -0.12 & -5.61 & 3.44 & 4.10 & 0.85 & -3.94 & 3.17 & -3.51 & 0.62 & \mathbf{7.03} & -5.15 \\ 4.11 & 1.23 & -4.84 & 0.19 & -3.60 & 2.16 & 3.36 & -0.01 & -4.31 & 0.17 & 3.97 & -2.02 & 2.49 & -0.04 & -4.11 & 2.57 \\ 3.72 & 2.20 & -3.51 & 1.67 & -5.30 & -0.01 & 4.53 & -2.49 & -3.30 & 0.32 & 2.06 & -4.40 & 3.14 & -1.10 & \mathbf{-6.54} & 1.40 \\ -2.85 & 1.72 & 5.21 & -0.32 & 4.97 & 1.00 & -5.68 & -1.24 & 3.87 & -1.45 & -5.40 & 1.67 & -4.30 & 1.20 & 3.73 & -3.39 \end{array}\right]$$

group 0: $\max - \min = \mathbf{13.52}$ | group 1: $\max - \min = \mathbf{13.58}$.

Every one of the 128 entries now sits in a tight band $[-6.79, 7.03]$, the two groups are within 0.5% of each other in dynamic range, and the absolute range collapses from 37.85 (Step 1) to 13.58 — a $2.8\times$ reduction. The importance metric also equalizes exactly: Lemma 1 applied to Λ_Q predicts every diagonal entry of $H_{\text{Had}}^\top \Lambda_Q H_{\text{Had}}$ collapses to the constant $\text{tr}(\Lambda_Q)/d = 443/128 \approx 3.46$, regardless of how peaky Λ_Q was. Table 10 confirms $\max_i (R_K^\top C_Q R_K)_{ii} / \text{mean}_i(\cdot) = 1.00$ exactly: *the* $46.9\times$ *peak in the* $Q^\top Q$ *importance is compressed to* $1.00\times$. This is the key invariant Step 2 produces; the per-group dynamic range alone is not yet better than pure Hadamard (compare to Step 5 below).

Step 3: $\tilde{K} = KU_Q H_{\text{Had}} P_K$ (permuted bit-reversal). P_K is a permutation matrix that reorders the 128 output channels. After $U_Q H_{\text{Had}}$ each Walsh output is a ±1-weighted sum of all eigenchannels, but the variances of those sums are not identical (driven by which Walsh row a high-eigenvalue eigvec activates most strongly in). PBR places the k-th largest eigenvector at Walsh output index $\beta(k)$ (the bit-reversal of k). At $G_K = 64$ this guarantees the top-2 eigenvectors split into the two groups (one each), the top-4 distribute one-per-quartile, and so on for any power-of-two grouping, so no single group inherits a disproportionate share of the high-energy eigenchannels.

$$(KU_Q H_{\text{Had}} P_K)_{t,\cdot} = \left[\begin{array}{rrrrrrrrrrrrrrrr} 2.24 & -3.71 & -2.86 & 3.72 & -0.87 & 4.11 & 1.90 & -2.85 & -3.72 & 4.10 & 1.52 & -3.30 & 1.91 & -4.31 & -2.31 & 3.87 \\ -5.76 & 1.12 & 4.37 & -5.30 & 5.12 & -3.60 & -2.58 & 4.97 & 3.19 & -3.51 & -3.09 & 3.14 & -1.67 & 2.49 & -0.07 & -4.30 \\ -3.73 & 6.89 & 6.73 & -3.51 & 4.12 & -4.84 & -6.06 & 5.21 & 5.55 & -3.94 & -4.94 & 2.06 & -4.91 & 3.97 & 5.04 & -5.40 \\ 4.08 & -5.61 & -5.77 & 4.53 & -6.79 & 3.36 & 4.18 & -5.68 & -4.16 & \mathbf{7.03} & 4.34 & \mathbf{-6.54} & 4.47 & -4.11 & -2.17 & 3.73 \\ \hline 1.08 & 2.31 & 0.66 & 2.20 & 0.64 & 1.23 & -1.30 & 1.72 & 0.29 & 0.85 & -0.05 & 0.32 & -2.08 & 0.17 & 0.88 & -1.45 \\ 0.92 & -0.12 & 1.49 & -0.01 & 1.16 & 2.16 & 3.27 & 1.00 & -3.70 & 0.62 & 2.28 & -1.10 & 1.97 & -0.04 & -1.10 & 1.20 \\ 0.26 & -1.79 & 1.83 & 1.67 & -3.18 & 0.19 & -1.90 & -0.32 & -2.67 & 3.17 & 3.97 & -4.40 & 4.21 & -2.02 & -2.61 & 1.67 \\ -3.39 & 3.44 & 0.72 & -2.49 & 0.97 & -0.01 & -3.10 & -1.24 & 1.80 & -5.15 & -0.37 & 1.40 & -0.63 & 2.57 & 4.09 & -3.39 \end{array}\right]$$

group 0: $\max - \min = \mathbf{13.82}$ | group 1: $\max - \min = \mathbf{9.36}$.

The *set* of 128 values is identical to Step 2 — only column-index positions are shuffled — but P_K has gathered most of the high-magnitude entries ($\{7.03, -6.54, -6.79, -6.06, 6.89, 6.73, \ldots\}$) into group 0 (rows 0–3) and left group 1 (rows 4–7) with a much tighter $[-5.15, 4.21]$ range. On this particular token PBR has increased one group's dynamic range slightly ($13.52 \to 13.82$) while shrinking the other's ($13.58 \to 9.36$); the residual on the tighter group drops as $(9.36/13.58)^2 \approx 0.48$ while the looser group is essentially unchanged. Averaged over all $T = 8000$ tokens, this gathering effect reduces the mean per-group max–min from 13.40 ($U_Q H_{\text{Had}}$) to 11.58 (OSCAR), translating to

a $1.23\times$ reduction in $\mathrm{tr}(E_K)$ (208 $\rightarrow$ 169). At this layer, ***PBR carries most of OSCAR's quantization-residual improvement*** over $U_Q H_{\mathrm{Had}}$ alone; the role of $U_Q H_{\mathrm{Had}}$ is to make Lemma 1 fire so that the importance metric is equalized exactly ($\max/\mathrm{mean} = 1.00$).

Degenerate case ($G_K = d$). pure Hadamard, QuaRot-style. A natural simpler alternative is to drop U_Q entirely and rotate by H_{Had} alone:

$$(KH_{\mathrm{Had}})_{t,\cdot} = \left[\begin{array}{cccccccccccccccc} 2.20 & -0.20 & 3.44 & 2.25 & -1.33 & 2.33 & 3.83 & 2.71 & -2.34 & -5.96 & 4.20 & 5.20 & -3.46 & -0.77 & 4.42 & 4.48 \\ \mathbf{7.15} & 5.23 & -4.17 & -3.54 & 5.85 & 5.28 & -3.61 & -1.41 & 1.05 & 3.94 & -1.22 & -2.85 & 1.48 & 1.03 & 2.05 & -0.46 \\ 1.02 & -1.19 & -2.61 & -2.01 & 0.93 & 3.00 & -1.01 & -3.19 & 3.09 & 2.20 & -5.85 & -5.58 & 4.90 & 3.06 & -4.39 & -3.55 \\ -3.07 & -3.65 & 0.48 & 4.83 & -3.59 & -2.92 & 3.38 & 5.09 & -2.81 & -0.89 & 1.22 & 0.46 & -3.49 & -2.21 & 2.88 & 0.84 \\ \hline -1.53 & 1.03 & 0.94 & 0.19 & -2.30 & 0.20 & 5.16 & 1.89 & -1.37 & -3.52 & 4.74 & 4.87 & -3.51 & -4.25 & 4.14 & 5.80 \\ 5.48 & 5.35 & -4.44 & -1.88 & 7.12 & 5.46 & -1.44 & -2.71 & 1.37 & 3.77 & 0.45 & -2.59 & 3.21 & 2.03 & -0.67 & 0.81 \\ -1.55 & 2.51 & -2.80 & -3.34 & 0.29 & 1.81 & -2.03 & -0.88 & 5.68 & 4.70 & \mathbf{-6.89} & -5.69 & 1.38 & 1.07 & -4.79 & -2.27 \\ -3.94 & -4.43 & 2.41 & 5.23 & -2.12 & -5.19 & 2.62 & 2.32 & -1.78 & -2.21 & 0.40 & 0.43 & -0.79 & -1.48 & 1.86 & 2.20 \end{array}\right]$$

group 0: $\max - \min = \mathbf{13.12}$ | group 1: $\max - \min = \mathbf{14.01}$.

Hadamard mixing on its own balances the two groups (13.12 $\approx$ 14.01) and on this token gives a per-group dynamic range comparable to $U_Q H_{\mathrm{Had}}$ (Step 2). The crucial structural difference is in the importance metric: C_Q is not diagonal in the channel basis, so Lemma 1 does not apply to $H_{\mathrm{Had}}^\top C_Q H_{\mathrm{Had}}$. We measure $\max/\mathrm{mean} = 1.72$ rather than 1.00 — the importance *floor* is not equalized.

	what the quantizer sees on $\tilde{K} = KR_K$		*importance*	*INT2 residual*
Rotation R_K	$\max\lvert\tilde{K}\rvert$	per-group max–min ($G_K = 64$)	$\frac{\max_i (R_K^\top C_Q R_K)_{ii}}{\mathrm{mean}_i(\cdot)}$	$\mathrm{tr}(E_K)$
I (no rotation)	36.3	25.7	43.0	233
H_{Had} (pure Hadamard, QuaRot-style)	11.1	13.3	1.72	206
U_Q (eigenbasis alone)	29.4	22.4	46.9	**1354**
$U_Q\, H_{\mathrm{Had}}$	7.46	13.4	**1.00**	208
$U_Q\, H_{\mathrm{Had}}\, P_K$ (OSCAR)	7.46	**11.6**	**1.00**	**169**

Table 10: Layer 10 of Qwen3-4B-Thinking, kv-head 0, $d = 128$, INT2 per-group ($G_K = 64$, two groups per head), $T = 8000$ tokens. $C_Q = Q^\top Q$ averaged over GQA groups (production setting). $\max|\tilde{K}|$ is the largest entry across all $T \times d$ positions; *per-group max–min* is $\mathbb{E}_t\left[\max_{i\in g} \tilde{K}_{t,i} - \min_{i\in g} \tilde{K}_{t,i}\right]$ averaged over the two $G_K = 64$ groups. The importance column measures how peaky the rotated importance metric's diagonal is (1.00 = all channels equally important; Lemma 1 forces this when C_Q is pre-diagonalized by U_Q). $\mathrm{tr}(E_K)$ is the unweighted INT2 residual. At $G_K = d = 128$ (one group per head) the permutation P_K would be a no-op and OSCAR collapses to $U_Q H_{\mathrm{Had}}$; we deliberately use $G_K = 64$ here so the PBR step has a non-trivial effect. Bold entries are extreme values referenced in the text.

The two "what the quantizer sees" columns make the OSCAR advantage on quantization residual concrete. Pure Hadamard cuts the per-group max–min from 25.7 (raw) to 13.3 ($1.9\times$); OSCAR cuts it further to 11.6 ($2.2\times$ over raw). Since INT2 residual scales as the square of dynamic range, the $\mathrm{tr}(E_K)$ ratios should be $(25.7/11.6)^2 \approx 4.9$ and $(13.3/11.6)^2 \approx 1.3$, matching empirical $233/169 = 1.4$ and $206/169 = 1.2$ (the empirical ratios are smaller than the squared dynamic range because the per-token correlation structure of the residual matters; the relation is only proportional, not exact). PBR alone contributes the bulk of OSCAR's advantage over $U_Q H_{\mathrm{Had}}$ at this layer ($\mathrm{tr}(E_K)$: 208 $\rightarrow$ 169, $1.23\times$). The importance column tells the orthogonal story: only the $U_Q H_{\mathrm{Had}}$ and OSCAR rows achieve $\max/\mathrm{mean} = 1.00$, the exact diagonal equalization predicted by Lemma 1; pure Hadamard alone leaves it at 1.72, and raw / U_Q alone leave it at 43–47.

Summary. The three factors are not interchangeable, and each provides a guarantee the others cannot:

- U_Q exposes the importance spectrum by diagonalizing C_Q, so per-channel importance becomes exactly λ_j. By itself it makes per-group dynamic range *worse* on real activations because the K-variance gets pushed into low-importance directions, not flattened.
- H_{Had}, composed with U_Q, equalizes the importance-metric diagonal to a constant $\mathrm{tr}(C_Q)/d$ via Lemma 1 ($\max/\mathrm{mean} = 1.00$ exactly). Without the U_Q pre-step (pure Hadamard alone), the importance metric is not equalized ($\max/\mathrm{mean} = 1.72$ here), so quantization error in any single rotated channel can land disproportionately on a high-importance direction in C_Q depending on the layer/token distribution.

- P_K ensures that for any power-of-two group size $G_K < d$, the importance hierarchy is recursively balanced across groups. This is where most of OSCAR's empirical $\mathrm{tr}(E_K)$ reduction over $U_Q H_{\mathrm{Had}}$ comes from on real activations. When $G_K = d$ it is a column permutation within a single group and has no effect; the PBR step is strictly beneficial only for finer grouping.

The same three-factor construction defines $R_V = U_S H_{\mathrm{Had}} P_V$ on the value side, with $C_S = V^\top S^\top S V$ (the attention-score-weighted value covariance) in place of C_Q.

A.5 Scale Determination and Per-token Clipping Details

Quantization is applied after the composed rotations in Eq. (3). At decoding step t, the key and value rows are first mapped into the rotated coordinates

$$\widetilde{k}_t = k_t R_K, \qquad \widetilde{v}_t = v_t R_V.$$

Before quantization, the implementation applies token-wise percentile clipping separately to keys and values. Given clip ratios $\rho_K, \rho_V \in (0, 1]$, define

$$\tau_t^{(K)} = \mathrm{quantile}_{\rho_K}\left(\{|\widetilde{k}_{t,c}|\}_{c=1}^{d}\right), \qquad \tau_t^{(V)} = \mathrm{quantile}_{\rho_V}\left(\{|\widetilde{v}_{t,c}|\}_{c=1}^{d}\right).$$

The clipped rows are

$$\widetilde{k}_t^{\mathrm{clip}} = \mathrm{clip}\left(\widetilde{k}_t, -\tau_t^{(K)}, \tau_t^{(K)}\right), \qquad \widetilde{v}_t^{\mathrm{clip}} = \mathrm{clip}\left(\widetilde{v}_t, -\tau_t^{(V)}, \tau_t^{(V)}\right).$$

In the current implementation, the default clip ratios are calibrated offline from rotated calibration activations, yielding $(\rho_K, \rho_V) = (0.96, 0.92)$.

For keys, the default implementation uses asymmetric INT2 quantization, optionally with groups along the channel dimension. Let $b = 2$ and $q_{\max} = 2^b - 1 = 3$. For a group g of a clipped rotated key row, write

$$\widetilde{k}_{t,g}^{\mathrm{clip}} \in \mathbb{R}^{1 \times G_K},$$

where G_K is the key quantization group size. The implementation computes a dynamic min–max scale and zero point for each token and group:

$$a_{t,g}^{(K)} = \min_c \widetilde{k}_{t,g,c}^{\mathrm{clip}}, \qquad b_{t,g}^{(K)} = \max_c \widetilde{k}_{t,g,c}^{\mathrm{clip}},$$

$$s_{t,g}^{(K)} = \frac{b_{t,g}^{(K)} - a_{t,g}^{(K)}}{q_{\max}}, \qquad z_{t,g}^{(K)} = -\frac{a_{t,g}^{(K)}}{s_{t,g}^{(K)}}.$$

The quantized and dequantized key group is then

$$\mathcal{Q}^+\left(\widetilde{k}_{t,g}^{\mathrm{clip}}\right) = \mathrm{clip}\left(\left\lfloor \frac{\widetilde{k}_{t,g}^{\mathrm{clip}}}{s_{t,g}^{(K)}} + z_{t,g}^{(K)} \right\rfloor, 0, q_{\max}\right),$$

and

$$\mathcal{Q}\left(\widetilde{k}_{t,g}^{\mathrm{clip}}\right) = \mathcal{Q}^-\left(\mathcal{Q}^+\left(\widetilde{k}_{t,g}^{\mathrm{clip}}\right)\right) = s_{t,g}^{(K)}\left(\mathcal{Q}^+\left(\widetilde{k}_{t,g}^{\mathrm{clip}}\right) - z_{t,g}^{(K)}\right).$$

Thus, unlike a fixed Frobenius-norm scale, the key scale is computed dynamically from the clipped rotated key row, independently for each token and each group. In our default configuration, $G_K = 64$.

Values use the same affine asymmetric INT2 quantization rule. For a group g of a clipped rotated value row, write

$$\widetilde{v}_{t,g}^{\mathrm{clip}} \in \mathbb{R}^{1 \times G_V},$$

where G_V is the value quantization group size. The implementation computes

$$a_{t,g}^{(V)} = \min_c \widetilde{v}_{t,g,c}^{\mathrm{clip}}, \qquad b_{t,g}^{(V)} = \max_c \widetilde{v}_{t,g,c}^{\mathrm{clip}},$$

and

$$s_{t,g}^{(V)} = \frac{b_{t,g}^{(V)} - a_{t,g}^{(V)}}{q_{\max}}, \qquad z_{t,g}^{(V)} = -\frac{a_{t,g}^{(V)}}{s_{t,g}^{(V)}}.$$

The quantized and dequantized value group is

$$\mathcal{Q}^{+}\left(\widetilde{v}_{t,g}^{\mathrm{clip}}\right) := \mathrm{clip}\left(\left\lfloor \frac{\widetilde{v}_{t,g}^{\mathrm{clip}}}{s_{t,g}^{(V)}} + z_{t,g}^{(V)} \right\rceil, 0, q_{\max}\right),$$

and

$$\mathcal{Q}\left(\widetilde{v}_{t,g}^{\mathrm{clip}}\right) := s_{t,g}^{(V)}\left(\mathcal{Q}^{+}\left(\widetilde{v}_{t,g}^{\mathrm{clip}}\right) - z_{t,g}^{(V)}\right).$$

Finally, after quantization–dequantization in the rotated coordinates, the approximations are mapped back to the original coordinate system:

$$\widehat{k}_t = \mathcal{Q}\left(\widetilde{k}_t^{\mathrm{clip}}\right) R_K^{\top}, \qquad \widehat{v}_t = \mathcal{Q}\left(\widetilde{v}_t^{\mathrm{clip}}\right) R_V^{\top}.$$

A.6 Proof of Theorem 1

We prove the key-side statement first. The row-vector rotation convention gives the frozen-error surrogate

$$\widetilde{\mathcal{L}}_K(R_k) = \mathrm{tr}\left(R_k^{\top} C_Q R_k E_K\right), \qquad R_k^{\top} R_k = I_d,$$

where

$$C_Q = U_Q \Lambda_Q U_Q^{\top}, \qquad \Lambda_Q = \mathrm{diag}(\lambda_1, \ldots, \lambda_d), \qquad \lambda_1 \geq \cdots \geq \lambda_d \geq 0,$$

and

$$E_K = \mathrm{diag}(\mu_1, \ldots, \mu_d), \qquad \mu_1 \leq \cdots \leq \mu_d.$$

Here E_K is treated as a fixed frozen residual covariance matrix.

Let

$$R_k = U_Q Z, \qquad Z^{\top} Z = I_d.$$

Substituting into the objective gives

$$\begin{aligned}\widetilde{\mathcal{L}}_K(R_k) &= \mathrm{tr}\left(Z^{\top} U_Q^{\top} C_Q U_Q Z E_K\right) \\ &= \mathrm{tr}\left(Z^{\top} \Lambda_Q Z E_K\right).\end{aligned}$$

Writing $Z = (z_{ij})$, we have

$$\left(Z^{\top} \Lambda_Q Z\right)_{ii} = \sum_{j=1}^{d} \lambda_j z_{ji}^2.$$

Therefore,

$$\widetilde{\mathcal{L}}_K(R_k) = \sum_{i=1}^{d} \sum_{j=1}^{d} \mu_i \lambda_j z_{ji}^2. \tag{4}$$

Since Z is orthogonal, the matrix $W = (w_{ji})$ with $w_{ji} = z_{ji}^2$ is doubly stochastic. Hence Eq. (4) is a linear function over the Birkhoff polytope, and its minimum is attained at a permutation matrix.

Thus,

$$\min_{R_k^{\top} R_k = I_d} \widetilde{\mathcal{L}}_K(R_k) = \min_{\Pi \in \mathcal{P}_d} \mathrm{tr}\left(\Pi^{\top} \Lambda_Q \Pi E_K\right),$$

where $\mathcal{P}_d$ denotes the set of $d \times d$ permutation matrices. By the rearrangement inequality, since

$$\lambda_1 \geq \cdots \geq \lambda_d \qquad \text{and} \qquad \mu_1 \leq \cdots \leq \mu_d,$$

the minimum is achieved by pairing the largest λ_j with the smallest μ_i. This is achieved by the identity permutation. Therefore, one minimizer is

$$Z = I_d,$$

and hence

$$R_k = U_Q.$$

The value-side proof is identical. The value surrogate is

$$\widetilde{\mathcal{L}}_V(R_v) = \operatorname{tr}\big(R_v^\top C_S R_v E_V\big), \qquad R_v^\top R_v = I_d,$$

where

$$C_S = U_S \Lambda_S U_S^\top, \qquad \Lambda_S = \operatorname{diag}(\nu_1, \dots, \nu_d), \qquad \nu_1 \geq \cdots \geq \nu_d \geq 0,$$

and

$$E_V = \operatorname{diag}(\eta_1, \dots, \eta_d), \qquad \eta_1 \leq \cdots \leq \eta_d.$$

Again, E_V is treated as a fixed frozen residual covariance matrix.

Let

$$R_v = U_S Z_V, \qquad Z_V^\top Z_V = I_d.$$

Then

$$\begin{aligned}\widetilde{\mathcal{L}}_V(R_v) &= \operatorname{tr}\big(Z_V^\top U_S^\top C_S U_S Z_V E_V\big) \\ &= \operatorname{tr}\big(Z_V^\top \Lambda_S Z_V E_V\big).\end{aligned}$$

Writing $Z_V = (\zeta_{ij})$, we obtain

$$\widetilde{\mathcal{L}}_V(R_v) = \sum_{i=1}^{d}\sum_{j=1}^{d} \eta_i \nu_j \zeta_{ji}^2.$$

As before, (ζ_{ji}^2) is doubly stochastic. The minimum is therefore attained at a permutation matrix. Since

$$\nu_1 \geq \cdots \geq \nu_d \qquad \text{and} \qquad \eta_1 \leq \cdots \leq \eta_d,$$

the rearrangement inequality shows that the identity permutation is optimal. Thus one minimizer is

$$Z_V = I_d,$$

and therefore

$$R_v = U_S.$$

This proves both statements.

A.7 Justification of Surrogate Objectives in Theorem 1

To justify the surrogate objectives in Theorem 1, we start from the downstream attention errors induced by quantizing keys and values. For the key branch, the empirical logit distortion on the calibration set is

$$\ell_K(R_k) = \sum_{i=1}^{N}\sum_{j=1}^{i} \big[q_i (k_j - \hat{k}_j)^\top\big]^2,$$

where $\hat{k}_j = \mathcal{Q}(k_j R_k) R_k^\top$. Writing the rotated quantization residual as

$$e_j^{(K)}(R_k) := \mathcal{Q}(k_j R_k) - k_j R_k,$$

we have $\hat{k}_j - k_j = e_j^{(K)}(R_k) R_k^\top$, so that

$$\ell_K(R_k) = \sum_{j=1}^{N} e_j^{(K)}(R_k)\, R_k^\top C_j R_k\, e_j^{(K)}(R_k)^\top, \qquad C_j := \sum_{i=j}^{N} q_i^\top q_i.$$

To obtain a tractable objective, we approximate the position-dependent matrices C_j by the global query target covariance

$$C_Q = \frac{1}{N}\sum_{i=1}^{N} q_i^\top q_i.$$

Defining the frozen residual covariance (where R_k is fixed)

$$E_K := \sum_{j=1}^{N} e_j^{(K)}(R_k)^\top e_j^{(K)}(R_k),$$

the key-side frozen-error objective becomes, up to the constant normalization factor $\frac{1}{N}$,

$$\tilde{\mathcal{L}}_K(R_k) = \mathrm{tr}(R_k^\top C_Q R_k E_K).$$

Under a similar approximation, the value-side downstream error is heuristically represented by the frozen-error surrogate

$$\tilde{\mathcal{L}}_V(R_v) = \mathrm{tr}\big(R_v^\top C_S R_v E_V\big).$$

On long calibration sequences where materializing the full $T \times T$ attention matrix S is expensive, our implementation approximates $C_S^\star = V^\top S^\top S V$ with a streaming score-energy estimator that weights each value row $v_{t,h}$ of KV head h by $w_{t,h} = k_{t,h} C_{Q,h} k_{t,h}^\top$ and accumulates

$$C_S = \frac{1}{H_{\mathrm{kv}}} \sum_{h=1}^{H_{\mathrm{kv}}} \frac{\sum_t w_{t,h}\, v_{t,h}^\top v_{t,h}}{\sum_t w_{t,h}}.$$

A.8 Empirical Check of the Diagonal Residual Assumption

To evaluate how closely the deployed quantizer satisfies the diagonal residual assumption in Theorem 1, we measured the uncentered residual second moment $E = \mathbb{E}[\epsilon^\top \epsilon]$ in the rotated coordinates across all 36 layers of Qwen3-4B-Thinking-2507 using the deployed clipped, grouped, asymmetric INT2 path (measuring ϵ relative to the pre-clipping rotated activation so that clipping error is included). Table 11 reports the layer means of the off-diagonal Frobenius ratio $r = \|E - \mathrm{diag}(E)\|_F / \|E\|_F$, the diagonal squared-Frobenius energy fraction $1 - r^2$, and the max/mean diagonal ratio. Under the full OSCAR rotation (UHP), 98.4% of key residual energy and 95.5% of value residual energy lies on the diagonal, and the max/mean diagonal ratio drops to 1.27 and 1.09, supporting the diagonal surrogate as a reasonable first-order approximation while confirming that the residual depends on the rotation.

Table 11: Residual second-moment diagnostics of the deployed INT2 quantizer, averaged across all 36 layers of Qwen3-4B-Thinking-2507.

	Keys			**Values**		
Statistic	I	U	UHP	I	U	UHP
Off-diagonal ratio r	0.248	0.218	0.126	0.301	0.165	0.213
Diagonal energy $1 - r^2$	93.8%	95.2%	98.4%	90.9%	97.3%	95.5%
Max/mean diagonal	38.38	42.29	1.27	6.66	13.82	1.09

B Algorithm Flow and Serving Procedure

Algorithm 1 gives the concrete OSCAR procedure used in our experiments and SGLang implementation.

B.1 Grouped-Query Attention and Index Conventions

Let H_q and H_{kv} denote the numbers of query and KV heads, $g = H_q / H_{\mathrm{kv}}$ the GQA group size, and $\mathcal{G}_h$ the g query heads sharing KV head h. For each KV head h, we compute $C_{Q,h} = \frac{1}{Tg} \sum_{i \in \mathcal{G}_h} Q_i^\top Q_i$, average $C_{Q,h}$ across the H_{kv} heads of the layer to obtain C_Q, and similarly average the score-weighted value moments to obtain C_S (Table 12). One $d \times d$ key rotation R_K and one $d \times d$ value rotation R_V are stored per layer and shared across its KV heads, leaving the runtime GQA head mapping unchanged. Keys are rotated after RoPE, queries are rotated by R_K so that $q_t R_K (k_j R_K)^\top = q_t k_j^\top$, and R_V is folded into W_V and W_O. For the bit-reversal permutation P_{br} [28, 29], let $\beta(i)$ reverse the $\log_2 d$ binary bits of $i \in \{0, \ldots, d-1\}$; with eigenvectors sorted in descending eigenvalue order σ, P_{br} places coordinate $\sigma(i)$ at index $\beta(i)$.

Algorithm 1 OSCAR calibration, prefill, and decode

Hyperparameters: sink length S_0, recent window W, block size G=128, bit-width b=2
State: BF16 sink/recent cache, rotated BF16 staging cache, packed INT2 history cache
procedure Calibrate($\mathcal{D}$)
for each layer and KV head **do**
  $Q, K, V \leftarrow \mathrm{ForwardDump}(\mathcal{D})$
  $S \leftarrow \mathrm{softmax}_{\mathrm{row}}(QK^\top/\sqrt{d} + M)$
  $C_K \leftarrow \frac{1}{N}\sum_n q_n^\top q_n$
  $C_S \leftarrow \frac{1}{N} V^\top S^\top S V$
  $U_Q \leftarrow \mathrm{EigVec}(C_K), \quad U_S \leftarrow \mathrm{EigVec}(C_S)$
  $R_K \leftarrow U_Q H_{\mathrm{Had}} P_{\mathrm{br}}, \quad R_V \leftarrow U_S H_{\mathrm{Had}} P_{\mathrm{br}}$
  $c_K, c_V \leftarrow \mathrm{CalibrateClip}(KR_K, VR_V)$
end for
return $\{R_K, R_V, c_K, c_V\}$
end procedure
procedure Prefill(X)
$Q, K, V \leftarrow \mathrm{Forward}(X)$
$\widetilde{K} \leftarrow KR_K, \quad \widetilde{V} \leftarrow VR_V$ {rotate before prefill write}
$\mathrm{Cache}_{\mathrm{sink}} \leftarrow (K, V)_{[:S_0]}$
$\mathrm{Cache}_{\mathrm{recent}} \leftarrow (K, V)_{[-W:]}$
for $i = S_0$ **to** $|X| - W - 1$ **do**
  QUANTIZEANDWRITE($\widetilde{K}_i, \widetilde{V}_i, i$)
end for
end procedure
procedure DecodeStep(x_t)
$q_t, k_t, v_t \leftarrow \mathrm{Forward}(x_t)$
$\widetilde{k}_t \leftarrow k_t R_K, \quad \widetilde{v}_t \leftarrow v_t R_V$
Append (k_t, v_t) to $\mathrm{Cache}_{\mathrm{recent}}$; append $(\widetilde{k}_t, \widetilde{v}_t)$ to rotated staging
if $|\mathrm{Cache}_{\mathrm{recent}}| > W$ **then**
  $(\widetilde{k}_j, \widetilde{v}_j) \leftarrow \mathrm{PopOldestRotatedStaging}()$
  QUANTIZEANDWRITE($\widetilde{k}_j, \widetilde{v}_j, j$)
end if
$\widehat{K}_{\mathrm{hist}}, \widehat{V}_{\mathrm{hist}} \leftarrow \mathrm{DequantHistory}(\mathrm{Cache}_{\mathrm{INT2}})$
$\widehat{K}_{\mathrm{hist}} \leftarrow \widehat{K}_{\mathrm{hist}} R_K^\top, \quad \widehat{V}_{\mathrm{hist}} \leftarrow \widehat{V}_{\mathrm{hist}} R_V^\top$
$K_{\mathrm{all}}, V_{\mathrm{all}} \leftarrow \mathrm{Concat}(\mathrm{Cache}_{\mathrm{sink}}, \widehat{K}_{\mathrm{hist}}, \mathrm{Cache}_{\mathrm{recent}})$
$o_t \leftarrow \mathrm{softmax}(q_t K_{\mathrm{all}}^\top/\sqrt{d} + M) V_{\mathrm{all}}$
return o_t
end procedure
function QuantizeAndWrite($\widetilde{k}, \widetilde{v}, i$)
$\bar{k} \leftarrow \mathrm{clip}(\widetilde{k}, c_K), \quad \bar{v} \leftarrow \mathrm{clip}(\widetilde{v}, c_V)$
$K_i^+ \leftarrow \mathrm{AffineINT2}(\bar{k}, G), \quad V_i^+ \leftarrow \mathrm{AffineINT2}(\bar{v}, G)$
$\mathrm{Cache}_{\mathrm{INT2}}[i] \leftarrow (K_i^+, V_i^+, \mathrm{scale}, \mathrm{zero})$
end function

C Related Work

KV-cache compression. The KV cache has become a major target for post-training compression because its memory footprint grows with both context length and batch size. One line of work reduces or reprioritizes cached content through eviction, budgeted retention, attention-sink preservation, or saliency-aware precision assignment, as in H2O, SnapKV, PyramidKV, StreamingLLM, and ZipCache [1, 47, 48, 18, 19, 49]. Other systems compress the cache through low-rank structure or cross-layer sharing, including GEAR, PALU, xKV, and MatryoshkaKV [50–53]. Another line keeps all tokens but lowers the precision of the cached tensors, which is the setting most closely related to OSCAR. KIVI, KVQuant, and WKVQuant study low-bit KV storage and identify channel-wise outliers, especially in keys, as a central obstacle to aggressive quantization [20, 4, 5]. These methods

Table 12: Shapes and conventions of calibration quantities (T: calibration tokens, d: head dimension).

Quantity	Shape	Convention
Q_i	$T \times d$	post-RoPE query head $i \in \mathcal{G}_h$
K_h, V_h	$T \times d$	post-RoPE key and value of KV head h
$C_{Q,h}, C_Q, C_S$	$d \times d$	calibration second-moment targets
R_K, R_V	$d \times d$	one pair per layer, shared across KV heads

show that KV quantization can be accurate, but they also leave open how far precision can be reduced without changing the serving layout.

2-bit KV quantization. Pushing KV cache quantization to 2 bits usually requires additional structure beyond naive rounding. SKVQ preserves a recent sliding window in higher precision and combines channel reordering with clipped dynamic quantization [21]. Kitty promotes sensitive key-cache channels to higher precision and combines this with a residual cache [6], while PM-KVQ uses progressive mixed precision and block-wise memory allocation for long-CoT models [54]. Related mixed-precision analyses also argue that keys should generally receive more precision than values under a fixed memory budget [55]. RotateKV uses outlier-aware adaptive rotations for 2-bit KV quantization [7]. These methods improve accuracy, but their use of channel-wise precision, residual buffers, progressive bit allocation, or adaptive layouts can complicate integration with paged KV-cache serving and fused decode kernels. OSCAR instead targets the same INT2 regime with fixed token-wise transforms and a uniform paged layout, so the quantization path can be fused into production serving.

Rotation-based quantization. Orthogonal transforms are an attractive way to make low-bit quantization more accurate without changing tensor shapes. QuaRot, HALO, and HOT show that rotations can reduce outlier concentration, redistribute activation energy, and improve low-bit rounding behavior [9, 56–58, 12], while DuQuant and MixQuant combine block rotations with channel permutations to balance outlier mass across blocks [59, 60]. In the KV-cache setting, TurboQuant develops a data-oblivious online vector-quantization method based on random rotations, per-coordinate Lloyd–Max quantization, and QJL-style residual coding [44]. KVLinC combines Hadamard rotation with lightweight linear correction to reduce attention errors under aggressive KV quantization [61]. OSCAR follows the same broad rotation principle, but asks which rotation target is appropriate for INT2 KV cache quantization. While [11] effectively mitigates outliers at 4-bit using learned rotations, it collapses at extreme INT2 levels because its rotations fail to align with the actual covariance structures consumed by downstream attention. Furthermore, it focuses primarily on theoretical algorithmic gains, lacking the custom kernels and system-level integration necessary for real-world deployment in modern serving frameworks.

Covariance-aware calibration. A related family uses calibration data or covariance information to choose quantization or compression directions. For weight–activation quantization, GPTQ and QuIP use Hessian information to guide low-bit rounding [31, 27], and WUSH derives closed-form adaptive transforms from weight and activation second moments [62]; for low-rank compression, Drone, ASVD, SVD-LLM, SVD-LLMv2, and CorDA use calibration activations or data-aware statistics to guide factorization [63–67]. On the KV-cache side, HaPPI, CARE, RecalKV, and CommonKV apply covariance- or Hessian-aware factorization to keys and values for KV-side projection or compression [68–71]. Post-training quantization methods such as SmoothQuant, AWQ, GPTQ, and OmniQuant also use calibration data to estimate scales, equivalent transformations, clipping thresholds, or second-order surrogates before deployment [30, 72, 31, 73]. OSCAR specializes this calibration paradigm to attention: it estimates target covariances induced by downstream attention ($Q^\top Q$ and score-weighted V), rather than raw-cache reconstruction moments ($K^\top K, V^\top V$), and uses them to derive fixed orthogonal post-RoPE rotations and clip thresholds for paged INT2 KV serving (Table 13).

D Discussion

Limitations. Our theoretical analysis explains why the proposed covariance-target rotations are optimal under a frozen-error surrogate with explicit assumptions (Appendix A), but it is not yet a full proof of optimality for the complete autoregressive decoding process. Tightening this gap—for example, by proving when the calibration estimator is unbiased or optimal for the true attention

Table 13: Comparison of OSCAR with representative low-bit quantizers.

Method	Tensors	Transform (statistic)	Granularity	Bits	Runtime layout / overhead
KIVI [20]	KV	none	K per-channel, V per-token	2	FP residual buffer
Kitty [6]	KV	none (channel sensitivity)	per-channel boost	≈2.4	mixed-width channels
TurboQuant [44]	KV	random rotation	per-coordinate Lloyd–Max	2.5–4	QJL residual
QuaRot [9]	W/A/KV	randomized Hadamard	per-token group	4	online Hadamard
SpinQuant [11]	W/A/KV	learned rotation (training)	per-token group	4	online Hadamard
DuQuant, MixQuant [59, 60]	W/A	rotation + permutation	per-channel/token	4	merged into weights
WUSH [62]	W/A	non-orthogonal (W, X moments)	block	4	fused block transform
OSCAR (ours)	KV	$UH_{\text{Had}}P_{\text{br}}$ ($Q^\top Q$, score-weighted V)	per-token group	2	fused into writes and W_V, W_O; paged

objective—is an important direction for future work. We also focus on INT2 KV-cache quantization and do not explore other numerical formats or mixed-format designs, such as NVFP4, different weight precisions, or weight/KV precision combinations, which may create different accuracy–systems trade-offs. Similarly, OSCAR uses a fixed bit-width for quantized history tokens and does not study per-token bit allocation or thought-adaptive token precision, which recent work such as ThinKV suggests may be useful for long reasoning traces [74]. On the systems side, our implementation targets SGLang-style paged serving on H100 GPUs; additional tuning is needed to understand the best kernel design on other hardware such as B200. Finally, OSCAR is orthogonal to channel-wise or mixed-precision methods: applying attention-aware/Hadamard rotations inside channel-wise schemes may further improve accuracy, but combining these ideas while preserving an efficient serving layout remains open.

Broader impact. OSCAR reduces KV-cache memory and serving cost, which can make long-context LLM inference more accessible and energy-efficient. At the same time, cheaper long-context inference may also lower the cost of deploying powerful generative models, including applications with misuse risks; OSCAR does not introduce new model capabilities, but should be deployed with the same safety and usage controls as the underlying LLMs.

E Additional Experimental Results

E.1 Additional Motivation Figures

Additional activation distributions across layers are shown in Figures 9 and 10.

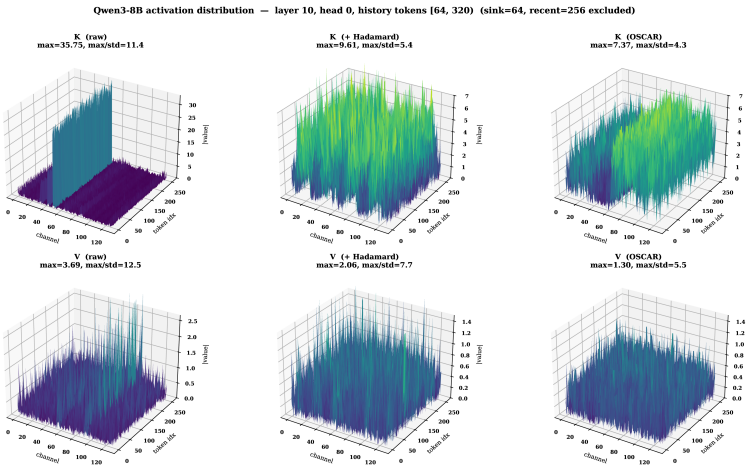

Figure 9: **Attention-aware rotations make history-token activations easier to quantize (Layer 10, Head 0).** Hadamard mixing flattens raw activation peaks by spreading energy across channels. OSCAR goes further: the target covariance separates directions that matter more or less to attention, and the Hadamard transform then mixes each part into a more uniform range.

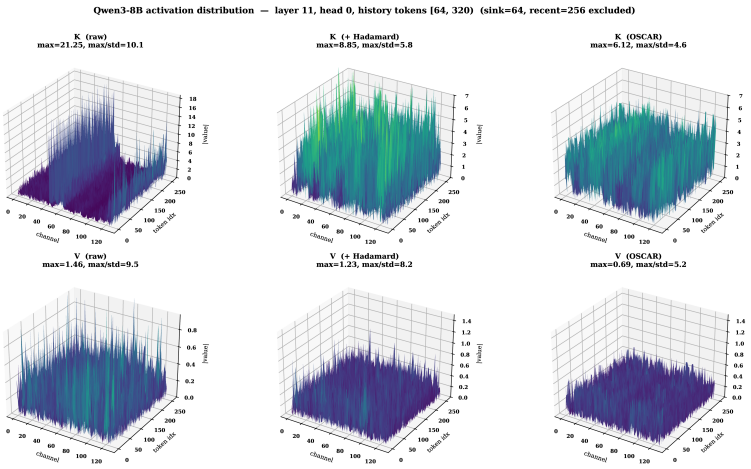

Figure 10: **Attention-aware rotations make history-token activations easier to quantize (Layer 11, Head 0).** Hadamard mixing flattens raw activation peaks by spreading energy across channels. OSCAR goes further: the target covariance separates directions that matter more or less to attention, and the Hadamard transform then mixes each part into a more uniform range.

E.2 Full Table For Main Accuracy Run

Table 14 reports the five independent runs for the Qwen3-4B-Thinking main OSCAR result. Table 15–17 report the five independent runs used to summarize Qwen3-8B OSCAR accuracy under three INT2 grouping configurations. Table 18–20 report the Qwen3-32B OSCAR runs. Table 21–23 report the GLM-4.7-FP8 detailed runs for Naive INT2, QuaRot/Hadamard rotation, and OSCAR.

Table 14: Full Qwen3-4B-Thinking results for the main OSCAR configuration (INT2 group size=128, sink=64, recent=256, $c_K = 0.96$, $c_V = 0.92$).

Task	Run 1	Run 2	Run 3	Run 4	Run 5	Mean	Std
GPQA	66.16	63.64	64.14	64.65	66.16	64.95	1.16
HumanEval	93.17	91.59	92.44	90.85	93.17	92.24	1.02
LiveCodeBench v6	45.03	43.27	45.61	48.54	44.44	45.38	1.97
AIME25	63.33	63.33	63.33	60.00	70.00	64.00	3.65
MATH500	92.59	92.79	92.99	93.19	92.18	92.75	0.39

Table 15: Full Qwen3-8B results for INT2 group=128.

Task	Run 1	Run 2	Run 3	Run 4	Run 5	Mean	Std
GPQA	53.03	54.04	54.55	57.58	55.05	54.85	1.70
HumanEval	86.59	88.54	88.90	87.56	87.68	87.85	0.91
LiveCodeBench v6	45.03	45.03	42.11	47.95	46.20	45.26	2.13
AIME25	66.67	63.33	56.67	63.33	63.33	62.67	3.65
MATH500	92.59	91.38	92.59	92.18	91.98	92.14	0.50

Table 16: Full Qwen3-8B results for INT2 group=64.

Task	Run 1	Run 2	Run 3	Run 4	Run 5	Mean	Std
GPQA	53.54	56.06	54.04	57.07	54.55	55.05	1.47
HumanEval	87.56	87.56	88.29	87.56	88.41	87.88	0.44
LiveCodeBench v6	47.37	44.44	49.12	45.03	45.61	46.32	1.91
AIME25	63.33	70.00	66.67	63.33	70.00	66.67	3.33
MATH500	92.38	92.79	91.58	91.18	93.19	92.22	0.83

Table 17: Full Qwen3-8B results for INT2 group=32.

Task	Run 1	Run 2	Run 3	Run 4	Run 5	Mean	Std
GPQA	52.02	55.56	59.09	52.53	56.57	55.15	2.93
HumanEval	87.93	87.32	88.05	88.78	88.66	88.15	0.59
LiveCodeBench v6	45.61	47.37	43.86	43.86	45.61	45.26	1.47
AIME25	66.67	53.33	63.33	66.67	60.00	62.00	5.58
MATH500	91.78	92.38	92.59	92.18	92.38	92.26	0.30

Table 18: Full Qwen3-32B results for OSCAR INT2 group=128.

Task	Run 1	Run 2	Run 3	Run 4	Run 5	Mean	Std
GPQA	59.60	59.09	57.58	60.10	–	59.09	1.09
HumanEval	90.37	89.76	89.27	89.76	89.27	89.68	0.45
LiveCodeBench v6	54.39	49.71	47.95	52.63	53.80	51.70	2.76
AIME25	63.33	66.67	76.67	76.67	70.00	70.67	5.96
MATH500	92.79	93.19	92.99	91.58	93.19	92.75	0.67

Table 19: Full Qwen3-32B results for OSCAR INT2 group=64.

Task	Run 1	Run 2	Run 3	Run 4	Run 5	Mean	Std
GPQA	60.61	63.64	61.11	62.12	54.55	60.40	3.47
HumanEval	89.88	89.76	89.39	91.22	90.37	90.12	0.71
LiveCodeBench v6	53.80	52.63	53.80	54.39	53.22	53.57	0.67
AIME25	76.67	70.00	80.00	66.67	76.67	74.00	5.48
MATH500	93.19	92.18	93.19	92.38	92.79	92.75	0.46

Table 20: Full Qwen3-32B results for OSCAR INT2 group=32.

Task	Run 1	Run 2	Run 3	Run 4	Run 5	Mean	Std
GPQA	63.64	57.07	56.57	60.10	59.60	59.40	2.83
HumanEval	92.20	90.85	89.76	90.73	90.24	90.76	0.91
LiveCodeBench v6	58.48	54.39	54.39	50.29	55.56	54.62	2.94
AIME25	66.67	70.00	70.00	66.67	63.33	67.33	2.79
MATH500	93.39	92.99	92.79	93.59	93.19	93.19	0.32

Table 21: Full GLM-4.7-FP8 results for the Naive INT2 baseline.

Task	Run 1	Run 2	Run 3	Mean	Std
GPQA	58.08	51.01	54.55	54.55	3.54
HumanEval	85.61	88.15	85.73	86.50	1.43
LiveCodeBench v6	33.33	44.44	33.33	37.03	6.41
AIME25	40.00	36.67	36.67	37.78	1.92
MATH500	87.40	85.80	86.60	86.60	0.80

Table 22: Full GLM-4.7-FP8 results for the QuaRot/Hadamard rotation baseline.

Task	Run 1	Run 2	Run 3	Mean	Std
GPQA	70.20	63.64	70.20	68.01	3.79
HumanEval	91.71	89.76	88.15	89.87	1.78
LiveCodeBench v6	47.37	53.80	44.44	48.54	4.79
AIME25	80.00	80.00	76.67	78.89	1.92
MATH500	90.20	90.20	90.80	90.40	0.35

Table 23: Full GLM-4.7-FP8 results for the OSCAR rotation group size=128 configuration.

Task	Run 1	Run 2	Run 3	Mean	Std
GPQA	73.23	74.24	73.23	73.57	0.58
HumanEval	90.85	91.34	90.98	91.06	0.25
LiveCodeBench v6	53.80	52.05	52.05	52.63	1.01
AIME25	76.67	80.00	80.00	78.89	1.92
MATH500	93.79	94.99	95.19	94.66	0.76

E.3 Ablation Detail Runs

Table 24 reports the individual runs behind the rotation-decomposition study in Table 4.

Table 24: Full individual runs for the rotation-decomposition study on Qwen3-8B.

Configuration	Task	Run 1	Run 2	Run 3	Mean	Std
Full OSCAR	GPQA	56.06	54.55	57.07	55.89	1.27
	HumanEval	87.93	87.80	88.17	87.97	0.19
	LiveCodeBench v6	43.86	47.37	46.78	46.00	1.88
	AIME25	66.67	70.00	66.67	67.78	1.92
	MATH500	92.18	92.38	92.59	92.38	0.20
Tensor-recon. target	GPQA	37.37	39.90	38.89	38.72	1.27
	HumanEval	35.49	36.22	34.88	35.53	0.67
	LiveCodeBench v6	4.09	4.09	2.92	3.70	0.68
	AIME25	13.33	16.67	10.00	13.33	3.33
	MATH500	64.73	63.73	64.53	64.33	0.53
w/o P_{br}	GPQA	53.54	54.04	51.01	52.86	1.62
	HumanEval	87.20	85.73	87.20	86.71	0.84
	LiveCodeBench v6	45.03	45.03	45.61	45.22	0.34
	AIME25	73.33	60.00	56.67	63.33	8.82
	MATH500	92.79	90.18	92.59	91.85	1.45
w/o H_{Had}	GPQA	50.00	53.54	50.51	51.35	1.91
	HumanEval	84.15	83.41	83.78	83.78	0.37
	LiveCodeBench v6	19.30	17.54	21.64	19.49	2.05
	AIME25	23.33	20.00	20.00	21.11	1.92
	MATH500	83.77	82.97	82.16	82.97	0.80
w/o U	GPQA	41.92	43.43	36.87	40.74	3.44
	HumanEval	37.68	39.88	38.05	38.54	1.18
	LiveCodeBench v6	4.09	4.09	5.26	4.48	0.68
	AIME25	13.33	16.67	16.67	15.56	1.92
	MATH500	65.33	63.33	65.73	64.80	1.29
No rotation	GPQA	18.69	21.72	15.15	18.52	3.29
	HumanEval	0.00	0.00	0.00	0.00	0.00
	LiveCodeBench v6	0.00	1.17	0.00	0.39	0.68
	AIME25	3.33	0.00	3.33	2.22	1.92
	MATH500	0.00	0.00	0.00	0.00	0.00

Table 25 reports the individual runs behind the sink/recent-window study in Table 5.

Table 25: Full individual runs for the sink/recent-window study on Qwen3-4B-Thinking-2507.

(S, R)	Task	Run 1	Run 2	Run 3	Run 4	Run 5	Mean	Std
$(0, 0)$	GPQA	0.00	0.00	0.00	–	–	0.00	0.00
	HumanEval	0.00	0.00	0.00	–	–	0.00	0.00
	LiveCodeBench v6	0.00	0.00	0.00	–	–	0.00	0.00
	AIME25	0.00	0.00	0.00	–	–	0.00	0.00
	MATH500	0.00	0.00	0.00	–	–	0.00	0.00
$(32, 128)$	GPQA	60.10	56.57	56.06	–	–	57.58	2.20
	HumanEval	91.95	92.07	91.71	–	–	91.91	0.19
	LiveCodeBench v6	40.35	43.27	38.60	–	–	40.74	2.36
	AIME25	56.67	60.00	50.00	–	–	55.56	5.09
	MATH500	93.39	92.99	91.58	–	–	92.65	0.95
$(64, 256)$	GPQA	66.16	63.64	64.14	64.65	66.16	64.95	1.16
	HumanEval	93.17	91.59	92.44	90.85	93.17	92.24	1.02
	LiveCodeBench v6	45.03	43.27	45.61	48.54	44.44	45.38	1.97
	AIME25	63.33	63.33	63.33	60.00	70.00	64.00	3.65
	MATH500	92.59	92.79	92.99	93.19	92.18	92.75	0.39
$(128, 512)$	GPQA	65.15	66.16	64.65	–	–	65.32	0.77
	HumanEval	93.29	92.80	93.41	–	–	93.17	0.32
	LiveCodeBench v6	45.03	45.61	45.03	–	–	45.22	0.33
	AIME25	70.00	66.67	66.67	–	–	67.78	1.92
	MATH500	93.39	93.19	93.39	–	–	93.32	0.12
$(256, 1024)$	GPQA	66.16	65.15	63.64	–	–	64.98	1.27
	HumanEval	92.20	93.66	93.54	–	–	93.13	0.81
	LiveCodeBench v6	45.03	48.54	45.03	–	–	46.20	2.03
	AIME25	66.67	70.00	66.67	–	–	67.78	1.92
	MATH500	93.19	93.39	93.39	–	–	93.32	0.12

Table 26 reports the per-task results behind the clip-threshold grid in Table 6.

Table 27 reports the individual runs behind the calibration-data regime study in Table 7.

Table 26: Full per-task results for the clip-threshold sweep on Qwen3-4B-Thinking-2507. Each row is one (c_K, c_V) setting with one run per task.

c_K	c_V	GPQA	HumanE	LCB v6	AIME25	MATH500	Mean
0.88	0.88	59.09	91.83	39.18	53.33	92.59	67.20
0.88	0.92	62.12	91.22	39.77	56.67	91.58	68.27
0.88	0.96	57.07	91.10	42.69	56.67	92.99	68.10
0.88	0.98	62.12	91.95	37.43	53.33	92.99	67.56
0.88	1.00	57.58	91.59	46.78	53.33	91.98	68.25
0.92	0.88	61.62	93.17	41.52	56.67	92.79	69.15
0.92	0.92	62.12	91.22	43.86	60.00	92.79	70.00
0.92	0.96	64.14	92.32	46.20	56.67	92.99	70.46
0.92	0.98	65.15	93.90	48.54	53.33	92.79	70.74
0.92	1.00	64.65	93.90	46.78	53.33	92.99	70.33
0.96	0.88	61.62	92.80	39.77	56.67	92.99	68.77
0.96	**0.92**	**64.14**	**93.54**	**46.20**	**56.67**	**92.38**	**70.59**
0.96	0.96	63.13	91.59	43.27	63.33	92.18	70.70
0.96	0.98	62.12	92.68	40.35	53.33	92.38	68.17
0.96	1.00	62.12	91.59	42.11	56.67	93.19	69.14
0.98	0.88	55.56	91.71	37.43	50.00	91.38	65.22
0.98	0.92	55.05	91.59	36.84	53.33	91.98	65.76
0.98	0.96	48.48	90.37	36.84	53.33	91.58	64.12
0.98	0.98	55.56	91.71	37.43	46.67	92.59	64.79
0.98	1.00	56.57	91.34	39.77	56.67	92.18	67.31
1.00	0.88	48.48	89.88	30.99	40.00	92.38	60.35
1.00	0.92	44.95	90.98	32.16	36.67	91.18	59.19
1.00	0.96	45.45	90.37	30.41	33.33	91.18	58.15
1.00	0.98	43.43	90.73	32.16	36.67	90.18	58.63
1.00	1.00	47.47	90.73	30.41	23.33	88.18	56.02

Table 27: Full individual runs for the calibration-data regime study on Qwen3-4B-Thinking-2507.

Calibration setting	Task	Run 1	Run 2	Run 3	Mean	Std
MMLU 2k	GPQA	61.62	58.08	62.12	60.61	2.20
	HumanEval	92.68	92.44	92.44	92.52	0.14
	LiveCodeBench v6	37.43	39.18	42.11	39.57	2.36
	AIME25	50.00	56.67	60.00	55.56	5.09
	MATH500	93.39	93.39	92.59	93.12	0.46
MMLU 8k	GPQA	62.63	58.59	60.10	60.44	2.04
	HumanEval	91.22	92.68	92.32	92.07	0.76
	LiveCodeBench v6	40.94	41.52	39.77	40.74	0.89
	AIME25	66.67	56.67	56.67	60.00	5.77
	MATH500	92.18	93.19	92.18	92.52	0.58
MMLU 16k	GPQA	63.64	64.65	64.14	64.14	0.51
	HumanEval	93.29	92.32	93.17	92.93	0.53
	LiveCodeBench v6	46.78	42.69	42.11	43.86	2.55
	AIME25	73.33	53.33	56.67	61.11	10.72
	MATH500	92.18	92.99	91.98	92.38	0.53
MMLU 32k	GPQA	62.63	61.62	59.60	61.28	1.54
	HumanEval	92.07	92.56	91.95	92.20	0.32
	LiveCodeBench v6	48.54	45.61	43.86	46.00	2.36
	AIME25	53.33	63.33	56.67	57.78	5.09
	MATH500	92.59	91.98	92.18	92.25	0.31
WikiText 8k	GPQA	61.11	60.61	61.62	61.11	0.51
	HumanEval	93.41	92.20	92.56	92.72	0.63
	LiveCodeBench v6	39.77	43.86	42.69	42.11	2.11
	AIME25	56.67	56.67	63.33	58.89	3.85
	MATH500	93.19	92.59	92.18	92.65	0.50
GPQA-Diamond 8k	GPQA	63.13	64.14	61.62	62.96	1.27
	HumanEval	92.56	92.56	92.56	92.56	0.00
	LiveCodeBench v6	46.20	44.44	43.27	44.64	1.47
	AIME25	63.33	63.33	60.00	62.22	1.92
	MATH500	92.38	92.59	92.99	92.65	0.31

E.4 Rotation Factor, Permutation, and Window Diagnostics

Pure eigenbasis (U-only) and post-Hadamard permutations. Table 28 evaluates applying the covariance eigenbasis U alone (without H_{Had} or P_{br}) on Qwen3-4B-Thinking-2507: GPQA drops from 64.95 ± 1.16 ($UH_{\mathrm{Had}}P_{\mathrm{br}}$) to 30.98 ± 1.27 and ΔPPL increases from $+0.063$ to $+0.477$, confirming that U must be paired with Hadamard mixing to equalize channel dynamic ranges for 4-level quantization. Table 29 holds U, H_{Had}, clipping, and windows fixed at $G{=}64$ and compares five channel-to-group permutations after UH_{Had}, including adaptations of the ZigZag [59] and MassDiff [60] grouping heuristics. Bit-reversal achieves the lowest token-logit KL, while all five permutations lie within 0.06 perplexity of one another.

Table 28: Pure-U vs. full $UH_{\mathrm{Had}}P_{\mathrm{br}}$ on Qwen3-4B-Thinking-2507 (WikiText-2 simulated, 2048 tokens; GPQA over 3 runs for pure U and 5 seeds for OSCAR/BF16).

Rotation	PPL	ΔPPL	KL	GPQA
BF16	11.440	–	0	67.27±1.80
U only	11.917	+0.477	0.0323	30.98±1.27
$UH_{\mathrm{Had}}P_{\mathrm{br}}$	**11.503**	**+0.063**	**0.0160**	**64.95±1.16**

Table 29: Post-UH_{Had} permutations on Qwen3-4B-Thinking-2507 ($G{=}64$, WikiText-2 simulated; BF16 PPL = 11.440).

Permutation after UH_{Had}	PPL	KL
none (UH_{Had})	11.552	0.01391
random	**11.535**	0.01303
ZigZag-style [59]	11.595	0.01374
MassDiff-style [60]	11.547	0.01253
bit reversal (OSCAR)	11.585	**0.01117**

One-sided windows and positional attention shift. Table 30 decouples the sink and recent windows at a fixed 320-token BF16 budget on Gemma-4-12B-it: sink-only protection $(320, 0)$ collapses to 0.00 on both AIME25 and LiveCodeBench v6, whereas recent-only protection $(0, 320)$ remains functional (38.33 and 50.86), showing that preserving recent reasoning tokens is the primary requirement during long generation. Table 31 measures one-step decode attention on Qwen3-8B at 4K context over eight blocks: the $(64, 256)$ windows lower attention KL from 0.551 to 0.186, while quantization noise slightly increases attention mass on the middle INT2 history ($+0.054$) rather than biasing attention toward the clean recent window (-0.040).

Table 30: One-sided and enlarged BF16 windows on Gemma-4-12B-it with OSCAR INT2.

(S_0, W)	AIME25	LCB v6	Seeds
BF16	54.17±0.72	62.86±0.99	3
$(320, 0)$ sink only	0.00	0.00	1
$(0, 320)$ recent only	38.33	50.86	1
$(32, 288)$	40.00	52.57	1
$(64, 256)$ default	40.14±2.44	54.10±1.84	3
$(64, 512)$	**48.19±2.77**	**60.95±0.66**	3

Table 31: Attention KL and change in attention mass (quantized minus FP16) by position bin on Qwen3-8B at 4K context.

Policy	Attn KL	Δsink	Δhist	Δrecent
default $(64, 256)$	**0.186**	-0.014	$+0.054$	-0.040
no windows	0.551	-0.023	$+0.025$	-0.003
sink only $(320, 0)$	0.448	-0.023	$+0.020$	$+0.002$
recent only $(0, 320)$	0.271	-0.013	$+0.055$	-0.042

E.5 Long-Context Diagnostics, Additional Models, and Baseline Comparisons

64K K/V isolation and large-model RULER. Table 32 isolates K-only and V-only quantization on Qwen3-8B after a 64K-token WikiText-2 prefix (512 scored tokens): V-only INT2 is virtually indistinguishable from BF16 (PPL 8.218 vs. 8.217, KL 0.0039), whereas K-only INT2 accounts for the full perplexity rise (23.318, KL 1.1371), confirming that long-context degradation stems from key-side logit error. On MiniMax-M2.7 (229B MoE, TP=4), paired evaluation across all 500 examples of RULER `niah_single_1` achieves 100.0% retrieval for both BF16 and OSCAR at 64K and 128K tokens, matching GLM-4.7 (97.2% vs. 97.2% at 128K in Table 3).

Table 32: K/V and window isolation on Qwen3-8B after a 64K-token prefix (simulated WikiText-2, 512 scored tokens).

Configuration	INT2 tensors	(S_0, W)	PPL	KL
BF16	–	–	8.217	0
V-only INT2	V	$(64, 256)$	8.218	0.0039
K-only INT2	K	$(64, 256)$	23.318	1.1371
K+V INT2 (default)	K, V	$(64, 256)$	21.862	1.0867
K+V INT2, recent=512	K, V	$(64, 512)$	21.712	1.0558
recent-only	K, V	$(0, 320)$	23.969	1.1483
sink-only	K, V	$(320, 0)$	47.492	1.8010
no windows	K, V	$(0, 0)$	57.190	1.9484

Broader model families, tool calling (BFCL), and reasoning-length truncation. Table 33 extends evaluation to six additional models spanning hybrid linear/full attention (Qwen3.5-4B using uniform INT2; Qwen3.5-35B-A3B using Lloyd–Max levels), sliding/global attention (Gemma-4-12B-it), large MoE (MiniMax-M2.7 at 229B, which also scores 70.8% on SWE-bench Verified under OSCAR INT2; and MiniMax-M3 at 428B under dense attention, with calibrated rotation on GPQA and Hadamard rotation on HumanEval/AIME25/MATH500), and 753B MLA (GLM-5.2 quantizing the 512-d latent c_{kv} with covariance rotation and Lloyd–Max while keeping k_{pe} in BF16). Table 34 reports multi-turn function calling on BFCL [75]. On GLM-5.2 GPQA at a 32K generation cap, INT2 lengthens median reasoning traces by $1.69\times$ and raises truncation from 14% to 28%, while option agreement among jointly answered questions is 97.1%, showing that larger drops on long-CoT tasks are primarily driven by budget truncation [45].

Table 33: BF16 → INT2 accuracy (%) on additional model families (3 seeds at 32K cap; Qwen3.5-35B-A3B AIME25: 8 seeds; MiniMax-M3 GPQA: 1 run; †MiniMax-M2.7 cells report single-seed OSCAR-only accuracy, with a 95K cap on AIME25/LCB and 70.8% on SWE-bench Verified).

Model	GPQA	HumanE	AIME25	MATH500	LCB v6	MMMU-Pro
Qwen3.5-4B	76.9→75.8	81.7→83.9	47.8→46.7	89.5→88.0	–	–
Qwen3.5-35B-A3B	83.3→84.0	83.9→86.6	66.7→62.1	92.8→91.7	–	–
Gemma-4-12B-it	63.47→60.94	93.70→93.09	54.17→40.14	93.73→90.13	62.86→54.10	42.89→40.67
MiniMax-M2.7†	80.3†	92.1†	90.0†	94.6†	68.4†	–
MiniMax-M3	88.38→85.35	93.1→91.7	77.8→67.8	79.7→78.7	–	–
GLM-5.2	81.3→68.7	91.5→88.2	76.7→75.6	94.8→93.8	–	–

Table 34: BFCL multi-turn accuracy (%), BF16 → INT2 (single runs).

Model	Overall	Long-ctx
Qwen3-8B	38.4→37.1	30.0→23.0
Qwen3-32B	49.1→44.4	44.5→36.0
Qwen3.5-4B	50.1→47.6	48.0→49.5

Comparisons with KIVI, Kitty, and TurboQuant, and upstream integration validation. Table 35 compares teacher-forced WikiText-2 perplexity and token-logit KL on Qwen3-8B (2048 scored tokens after 512-token prefixes and 512 scored tokens after 64K-token prefixes, matching OSCAR's BF16 windows on TurboQuant) alongside same-harness 4K-cap downstream evaluations using Kitty's official HF evaluation protocol (Table 36). On 5-shot GSM8K [76] (n=1319, 4K cap) on Qwen3-4B-Thinking-2507, TurboQuant's native mixed-precision vLLM preset (K3V3+NC with 4 BF16 boundary layers, 4.61 BPE) achieves $81.05/51.55$ strict/flexible accuracy versus $83.09/58.07$ for OSCAR in SGLang (3.55 stored BPE) and $87.19/61.87$ for BF16. Table 37 reports wall time for a fixed 64K-input, 32K-output decode benchmark on Qwen3-8B (batch size 1, $1\times$H100), and Table 38 verifies our fresh-upstream SGLang port [22] against the research fork reference on Qwen3-8B.

Table 35: Teacher-forced WikiText-2 PPL and token-logit KL on Qwen3-8B (simulated quantization).

Method	K/V	PPL	KL	PPL@64K
FP16	16/16	10.155	0	8.217
KIVI [20]	2/2	11.251	0.1316	–
KIVI* (32-tok sink)	2/2	10.623	0.0440	–
Kitty (12.5% K INT4)	2/2	10.311	0.0249	–
TurboQuant [44]	2/2	50.769	1.9467	973.368
TurboQuant [44]	4/2	11.810	0.1814	**9.967**
OSCAR	2/2	**10.167**	**0.0106**	21.862

Table 36: Same-harness 4K-cap evaluation on Qwen3-8B using Kitty's official protocol (single runs).

Method	GPQA	HumanE	AIME25
FP16	42.35	85.98	26.67
KIVI	33.16	75.61	13.33
KIVI*	46.43	84.76	20.00
Kitty	43.88	87.80	13.33
OSCAR	43.37	**87.80**	**20.00**

Table 37: Fixed 64K-input + 32K-output decode on Qwen3-8B (1×H100, batch size 1).

Method / Runtime	Wall time	Output tok/s
OSCAR (SGLang)	**5.43 min**	**100.65**
HF `generate` (FP16)	55.29 min	9.88
KIVI Qwen3 simulator	61.07 min	8.94
Kitty released path	93.27 min	5.86

Table 38: Upstream SGLang integration validation on Qwen3-8B (GPQA-Diamond 198 questions, 32K cap).

Implementation	GPQA	8.6K Needle	Failures
Research fork reference	56.06	–	0
Fresh-upstream port	**57.07**	**exact (8,659 tok)**	**0**

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and Section 1 state the contributions and their scope, including the remaining long-context gap on Qwen3 and the surrogate nature of the theory; Sections 3–5 support them, and Section 1 lists which components are adopted from prior work.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: Limitations are discussed in Appendix D, covering the surrogate scope of the theoretical analysis, numerical format and precision choices, hardware and serving-stack scope, and long-context behavior.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes].

Justification: Theorem 1 states its assumptions (positive semidefinite target, fixed diagonal residual with the stated ordering); the proof is in Appendix A.6, the approximations leading to the surrogate are in Appendix A.7, and Appendix A.8 checks the diagonal assumption empirically.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes].

Justification: Section 5.1 gives the models, tasks, decoding parameters, seeds, calibration data, and OSCAR hyperparameters; Appendix B gives the algorithm, GQA conventions, permutation index map, quantizer formulas, and bit accounting; Appendix E gives per-run results and the protocols of all additional experiments.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes].

   Justification: The SGLang implementation is available at https://github.com/FutureMLS-Lab/OSCAR, and the calibrated rotations at https://huggingface.co/Zhongzhu/OSCAR-RotationZoo; all evaluation data are public benchmarks.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: Section 5.1 specifies the models, benchmarks, generation settings, number of seeds, calibration data, and hardware; OSCAR involves no training, and its hyperparameters (window sizes, clip ratios, group sizes) are fixed as stated, with sensitivity studies in Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes].

Justification: Accuracy tables report the mean and sample standard deviation over five seeds (Qwen3) or three runs (GLM-4.7), which capture sampling randomness during generation; single-run results are marked as such, and Appendix E lists every run.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: The main experiments run on NVIDIA H100 80 GB GPUs (one GPU for the 4B and 8B models, two for Qwen3-32B, eight for GLM-4.7, four for MiniMax-M2.7), and the largest additional models in Appendix E.5 use two 8-GPU nodes; calibration is a single prefill pass over 8,878 tokens followed by $d \times d$ eigendecompositions. Appendix E.5 reports wall times for long decoding. The project also used compute for preliminary experiments that are not reported.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: The work evaluates publicly released models on public benchmarks to improve serving efficiency; we have reviewed the NeurIPS Code of Ethics and identify no deviation.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes].

    Justification: Appendix D discusses the lower cost and energy of long-context serving, the corresponding reduction in the cost of misuse, and the risk of quality changes that standard benchmarks miss.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: This paper does not release a pretrained model or dataset with high misuse risk; it proposes a KV-cache quantization method for existing models.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes].

Justification: We cite every model, benchmark, serving system, and baseline implementation we use (Section 5.1 and Appendix E.5); all are publicly released for research use under their respective licenses (e.g., Apache-2.0 for Qwen3, SGLang, and vLLM; MIT for HumanEval, GSM8K, and MATH; CC BY 4.0 for GPQA), and we use them only for evaluation.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes].

Justification: We release the SGLang implementation and the calibrated rotation matrices; their usage is documented in the linked repositories. We release no new dataset or model weights.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A].

Justification: The work does not involve crowdsourcing or human-subject experiments.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: The work does not involve human subjects and therefore does not require IRB approval.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A].

    Justification: The core method does not use LLMs as an original or non-standard component; LLMs are only the inference targets evaluated by the quantization method.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.